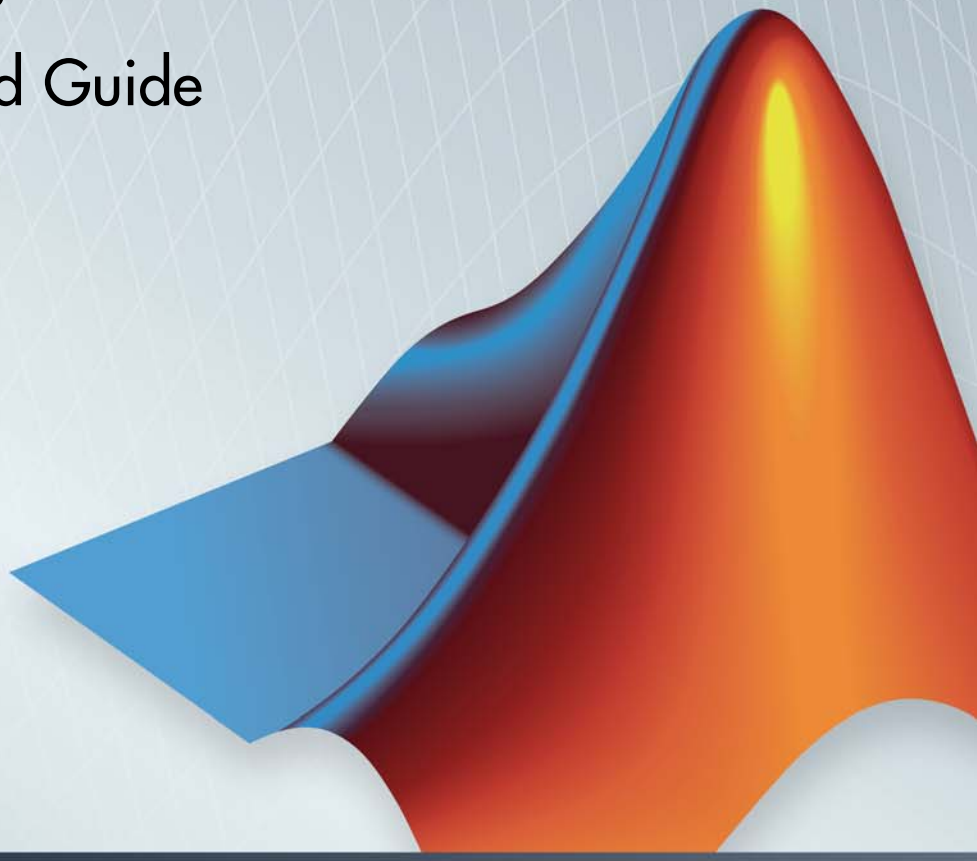
SimMechanics™

Getting Started Guide

R2011b



MATLAB® & SIMULINK®

MathWorks®

How to Contact MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

SimMechanics™ Getting Started Guide

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `www.mathworks.com/trademarks` for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see `www.mathworks.com/patents` for more information.

Revision History

October 2008	Online only	New for Version 3.0 (Release 2008b)
March 2009	Online only	Revised for Version 3.1 (Release 2009a)
September 2009	Online only	Revised for Version 3.1.1 (Release 2009b)
March 2010	Online only	Revised for Version 3.2 (Release 2010a)
September 2010	Online only	Revised for Version 3.2.1 (Release 2010b)
April 2011	Online only	Revised for Version 3.2.2 (Release 2011a)
September 2011	Online only	Revised for Version 3.2.3 (Release 2011b)

1

Introducing SimMechanics Software

SimMechanics™ software models, simulates, and visualizes mechanical systems, together with Simulink® and MATLAB®.

Product Overview

In this section...

Product Definition

SimMechanics software is a block diagram modeling environment for the engineering design and simulation of rigid multibody machines and their motions, using the standard Newtonian dynamics of forces and torques.

With SimMechanics software, you can model and simulate mechanical systems with a suite of tools to specify bodies and their mass properties, their possible motions, kinematic constraints, and coordinate systems, and to initiate and measure body motions. You represent a mechanical system by a connected block diagram, like other Simulink models. You can also incorporate hierarchical subsystems.

The visualization tools of SimMechanics software display and animate 3-D machine geometries, before and during simulation.

Mechanical Simulation and Physical Modeling

SimMechanics software is based on the Simscape™ software, the platform product for the Simulink Physical Modeling family, encompassing the modeling and design of systems according to basic physical principles. Simscape software runs within the Simulink environment and interfaces seamlessly with the rest of Simulink and with MATLAB. Unlike other Simulink blocks, which represent mathematical operations or operate on signals, Simscape blocks represent physical components or relationships directly.

Note This *SimMechanics Getting Started Guide* assumes that you already have some experience with modeling mechanical systems and with building and running models in Simulink.

Related Products

In this section...

Required Products

You must have current versions of the following products installed to use SimMechanics software:

- MATLAB
- Simulink
- Simscape

SimMechanics Visualization Requirements

The SimMechanics visualization window requires Silicon Graphics OpenGL® graphics support on your system in order to display and animate mechanical systems.

You can improve your speed and graphics resolution by adding a graphics accelerator hardware card to your system. Animation of simulations is sensitive to central processor and graphics card speed and memory. Experiment to find a reasonable compromise between quality and speed for your system.

STL Graphics Files for Customized Body Shapes. To switch from the SimMechanics visualization default body surface geometries to customized body shapes, you need a stereolithographic (STL) graphics file for each customized body. You can obtain these from computer-aided design (CAD) assemblies via the SimMechanics Link exporter, from graphical editors, or by manual creation.

Support for Recorded Animations

You can record simulation animations in Microsoft Audio Video Interleave® (AVI) format using the SimMechanics visualization. To play back AVI

files, you need an AVI-compatible media application. MATLAB has an internal movie player compatible with AVI. You can also use an external AVI-compatible player.

If you want to compress your SimMechanics AVI recordings, you need the Indeo 5 codec installed on your system to record them. Your AVI player might also require this codec to view compressed recordings.

Other Related Products

The related products listed on the SimMechanics product page at the MathWorks Web site include toolboxes and blocksets that extend the capabilities of MATLAB and Simulink. These products will enhance your SimMechanics experience in various applications.

SimMechanics Link Utility

The SimMechanics Link utility interfaces MATLAB with external mechanical applications such as CAD platforms. It generates Physical Modeling XML files that you can import to automatically generate SimMechanics models representing CAD assemblies. See the SimMechanics Link documentation.

Physical Modeling Product Family

Use the Physical Modeling product family to model physical systems in Simulink. In addition to SimMechanics software, it includes:

- Simscape, the platform and unifying environment for Physical Modeling products
- SimElectronics®, for modeling and simulating electronic systems
- SimDriveline™, for modeling and simulating drivetrain systems
- SimHydraulics®, for modeling and simulating hydromechanical systems
- SimPowerSystems™, for modeling and simulating electrical power systems

For More Information About MathWorks Products

For more information about any MathWorks® software products, see either

- The online documentation for that product if it is installed

- The MathWorks Web site at `www.mathworks.com`; see the “Products & Services” section.

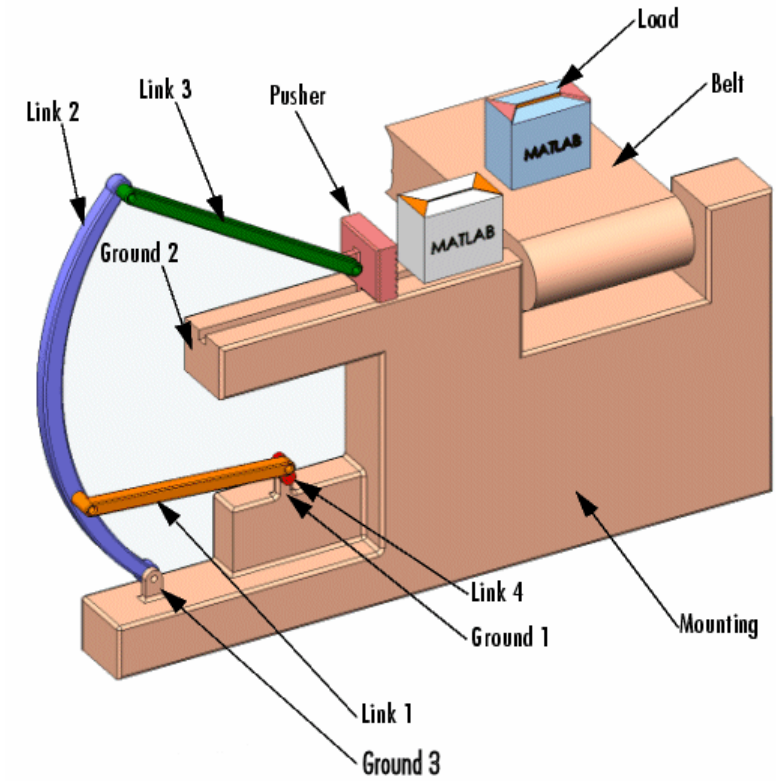
Simulate Conveyor Belt Model

In this section...

What the Example Represents

This example model uses a few blocks in the library to simulate a simple machine with feedback control. You will see how SimMechanics features build upon standard Simulink features to model a mechanical system.

The example model simulates a conveyor belt loading mechanism. A simple controller (not shown), with a sensor and an actuator, guides the mechanism with a saturation limit and anti-windup logic for the applied torque. You can adjust the controller and set the stopping point for the pusher.



Conveyor Loader Mechanism

What the Example Illustrates

The conveyor mechanism example illustrates some important SimMechanics features:

- Representing bodies and degrees of freedom with Body and Joint blocks, respectively
- Using SimMechanics blocks with normal Simulink blocks
- Feeding Simulink signals to and from SimMechanics blocks with Actuator and Sensor blocks, respectively
- Encapsulating groups of blocks into subsystems

- Visualizing and animating a mechanism by its component bodies

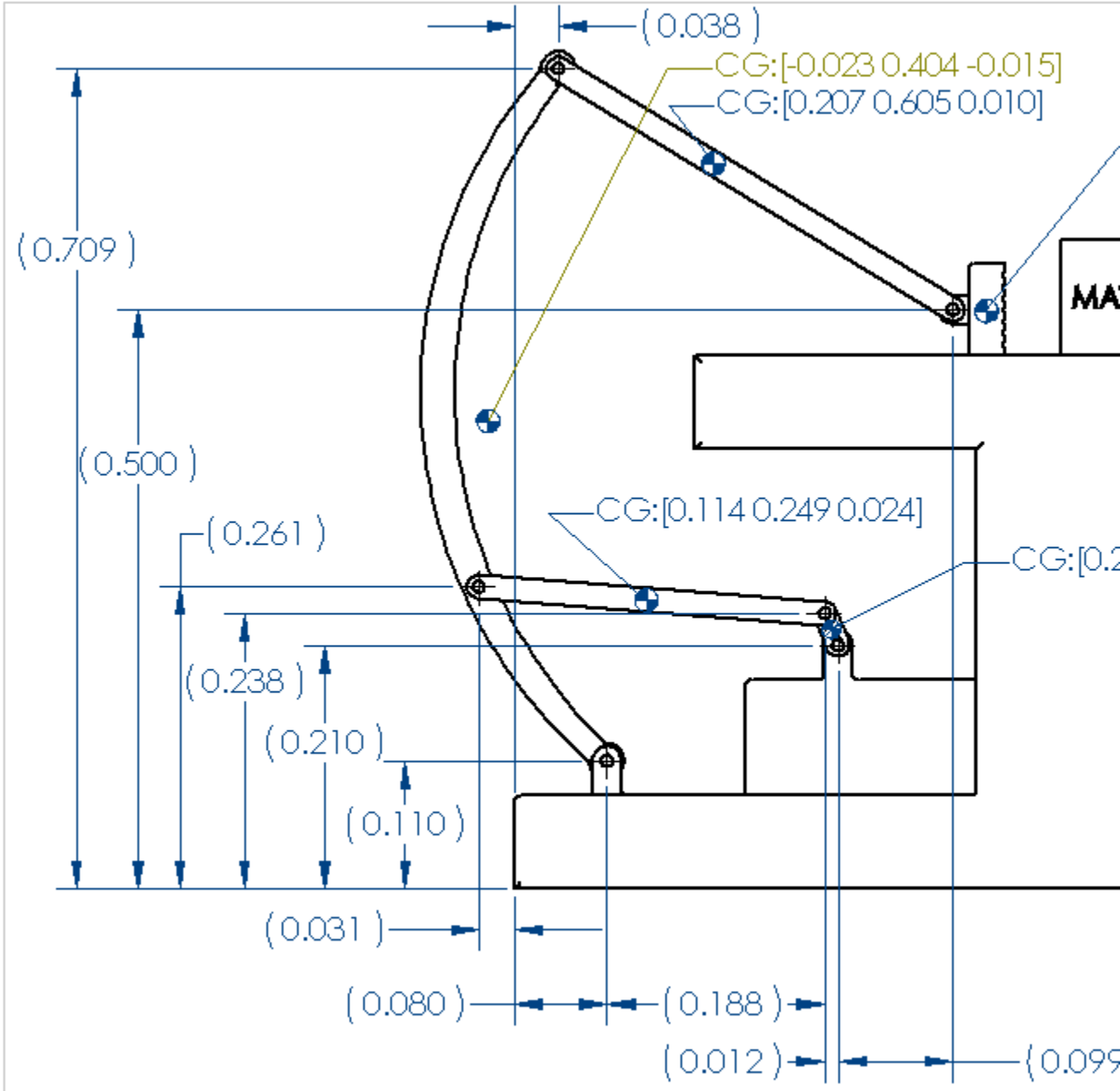
---

Caution You might want to make modifications to this example model. To avoid errors,

- Do not attempt to connect Simulink signal lines directly to SimMechanics blocks other than Actuators and Sensors.
- Keep the collocation of the Body coordinate system origins on either side of each assembled Joint to within assembly tolerances.

  Saving modified example models in a different folder from the examples is recommended.

The following figure shows a detailed schematic of the conveyor belt loading mechanism.


(0.038)
CG:[-0.023 0.404 -0.015]
CG:[0.207 0.605 0.010]
(0.709)
(0.500)
(0.261)
CG:[0.114 0.249 0.024]
(0.238)
(0.210)
(0.110)
(0.031)
(0.080)
(0.188)
(0.012)
(0.099
NOTE 1: ALL DIMENSIONS ARE IN METERS
DIMENSIONS ARE IN METE
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ±

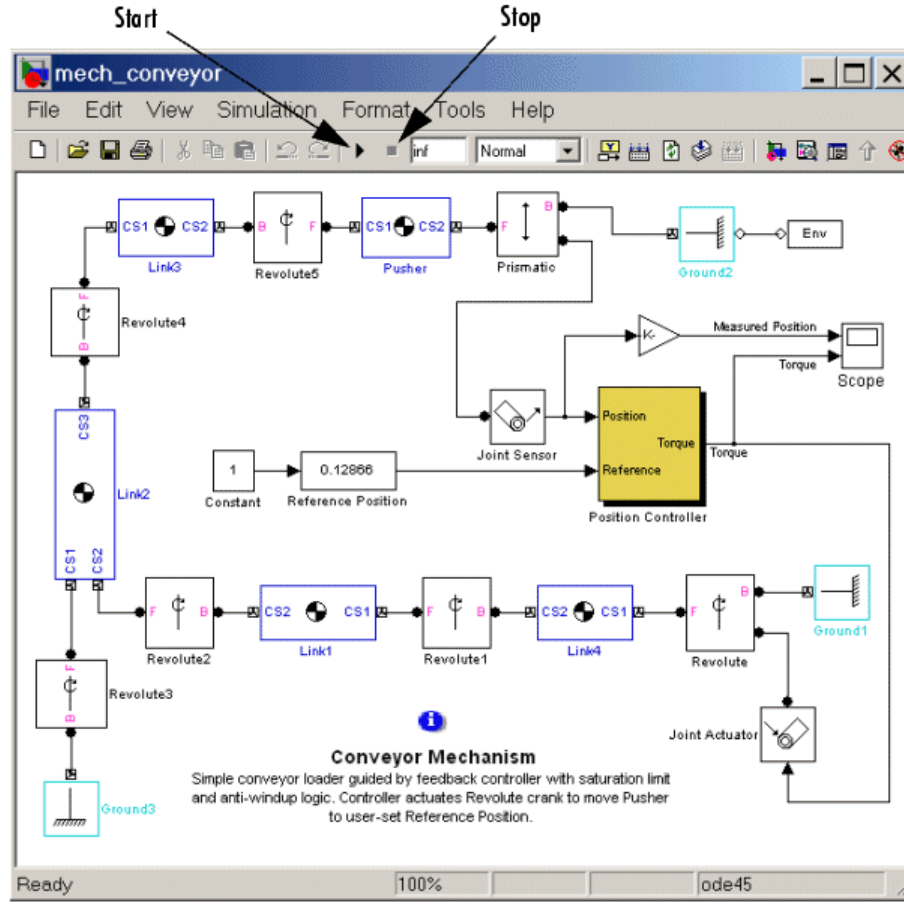
Opening the Model

To get started quickly with the conveyor example model, follow either of these steps:

- Enter `mech_conveyor` at the MATLAB command line.
- If you are working with the MATLAB Help browser, click the model name `mech_conveyor` here.

The Block Diagram Model

The block diagram model opens in a model window.



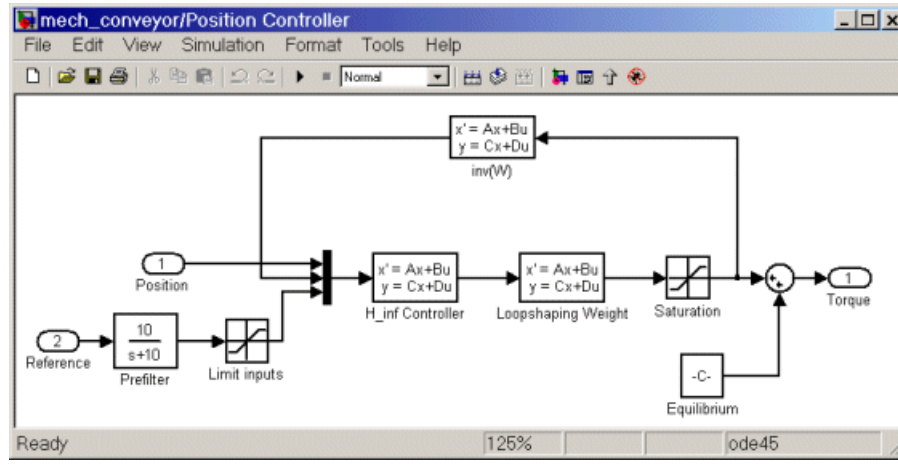
What the Model Contains

Here are some critical features of the model:

- Ignore the Position Controller, Joint Sensor, and Joint Actuator blocks for a moment. Note that the loading mechanism follows the tree of bodies and joints shown in the figure, Conveyor Loader Mechanism on page 1-7:

- There are four rotating link bodies and one sliding pusher body, as well as three ground points on the immobile mounting represented by Ground blocks. Double-click the Body and Ground blocks to see their dialog boxes.
- The pusher slides and the links rotate relative to one another and to the ground points on the mounting. There are seven apparent degrees of freedom (DoFs) in the system, represented by seven Joints, but the geometry constrains the motion to one actual DoF. Double-click the Revolute blocks to see how rotational DoFs are expressed in their dialog boxes.
- The Prismatic block expresses the linear motion of Pusher relative to Ground_2. The Revolute block expresses the angular motion of Link4 (the crank of the whole mechanism) relative to Ground_1.

- The Joint Sensor detects the position of Pusher via the Prismatic block. The Joint Actuator applies torque to Link4 via the Revolute block. Double-click the Sensor and Actuator blocks to view how the mechanical motions and forces/torques are transformed into Simulink signals.
- The Position Controller subsystem converts the Pusher position information into a feedback signal to actuate Revolute and thus Link4. You can open the Position Controller block to view this subsystem, which is made of normal Simulink blocks.



- The Reference Position block gives you control over the stopping position of the pusher by modulating the control signal that actuates Revolute. Maintaining the initial pusher position requires a fixed torque on Revolute.
- Open the Scope block. You can view both the Pusher position in millimeters (`mm`) relative to Ground_2 as the Measured Position plot and the torque in newton-meters (N-m) applied to Link4 relative to Ground_1 as the Torque plot.

Running the Model

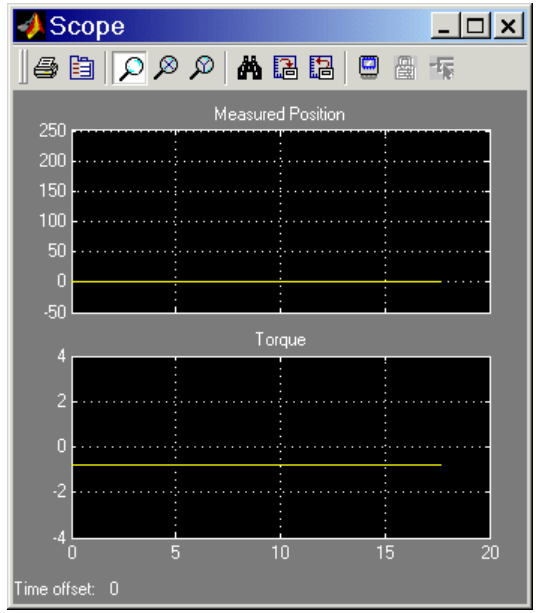
You can now run the model as it is when you first open it:

1. In the **Simulation** menu, select **Configuration Parameters**. The Configuration Parameters dialog box appears. Select the **Solver** node:
   a. The preset **Stop time** is `inf`, so the simulation keeps running once you start it. You should leave it at `inf` and stop the simulation manually the first few times you run it.

      Later you can apply a finite stop time (in seconds) if you want.

   b. Leave the **Solver options** entries at default values and close the box.
2. From the **Simulation** menu, select **Start**. In Microsoft Windows®, you can also click the **Start** button in the model window toolbar.

   The measured position of the pusher and the torque applied to maintain that position start and remain essentially constant in the Scope plots. The applied torque is adjusted to maintain the initial pusher position.



3 To see greater detail at the simulation start, stop the simulation before the time passes 20 seconds and zoom in on the Scope plots.

Modifying the Model

Here is a modification of the example you can try. It illustrates the simple controller that you can adjust to change the motion of the pusher.

To make these modifications, it is best to close and restart the example.

Changing the Pusher Reference Position

The Reference Position block is actually a Simulink Slider Gain block (from the Simulink Math Library) and controls where the pusher comes to rest.

You can adjust the Reference Position block to change where the pusher stops:

1 Open the Reference Position block. You see an adjustable slider to set the position of the pusher's rest point.

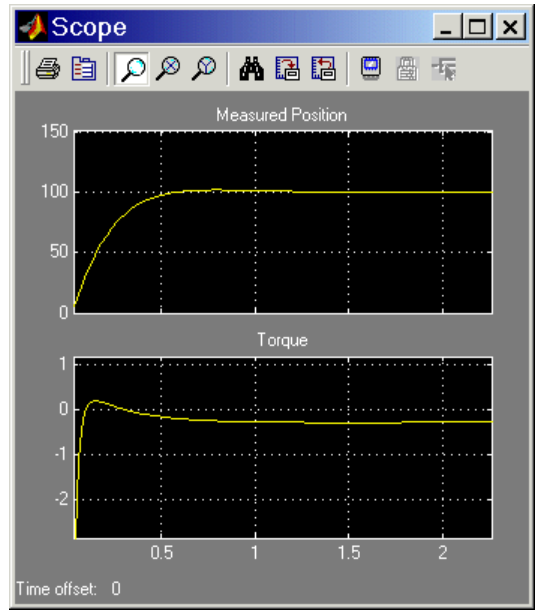
2. Enter values in the **Low** and **High** fields to set the lower and upper limits of the allowed slider range. The defaults in this example are `0` and `0.2`, with implied units of meters (`m`).

3. Enter a value in the central field to set the pusher stopping point, which you can also adjust by clicking and dragging the slider between the lower and upper limits. The default is `0` (meters).

You can apply changes to the reference position to the simulation in two ways:

- Reset the Reference Position block first, then start the example. You see the pusher trajectory track differently now, toward the new stopping point.

  For example, resetting the Reference Position to `0.1` and restarting the example produces these Scope plots, with **Autoscale** and zooming applied. The asymptotic measured position now tends to 100 mm (0.1 m), and the torque applied to keep the pusher there has changed:



- Start the example with the Reference Position block open and move the slider up and down as the simulation runs. Watch the Scope. The measured position and necessary torque change to follow the new reference position.

Visualizing and Animating the Model

Another modification you can make illustrates a powerful SimMechanics feature, visualization of a mechanism and animation of its simulated motion.

You can visualize and animate the conveyor mechanism by opening the SimMechanics visualization window. This window lets you display the bodies in two standard abstract forms:

- *Equivalent ellipsoids* use the inertia tensors and masses of the bodies. Each body has a unique homogeneous ellipsoid equivalent to it in mass and inertia tensor.
- *Convex hulls* use the attached Body coordinate systems (CSs) of the bodies to define a shape outlined by the CS origins.

Convex Hulls

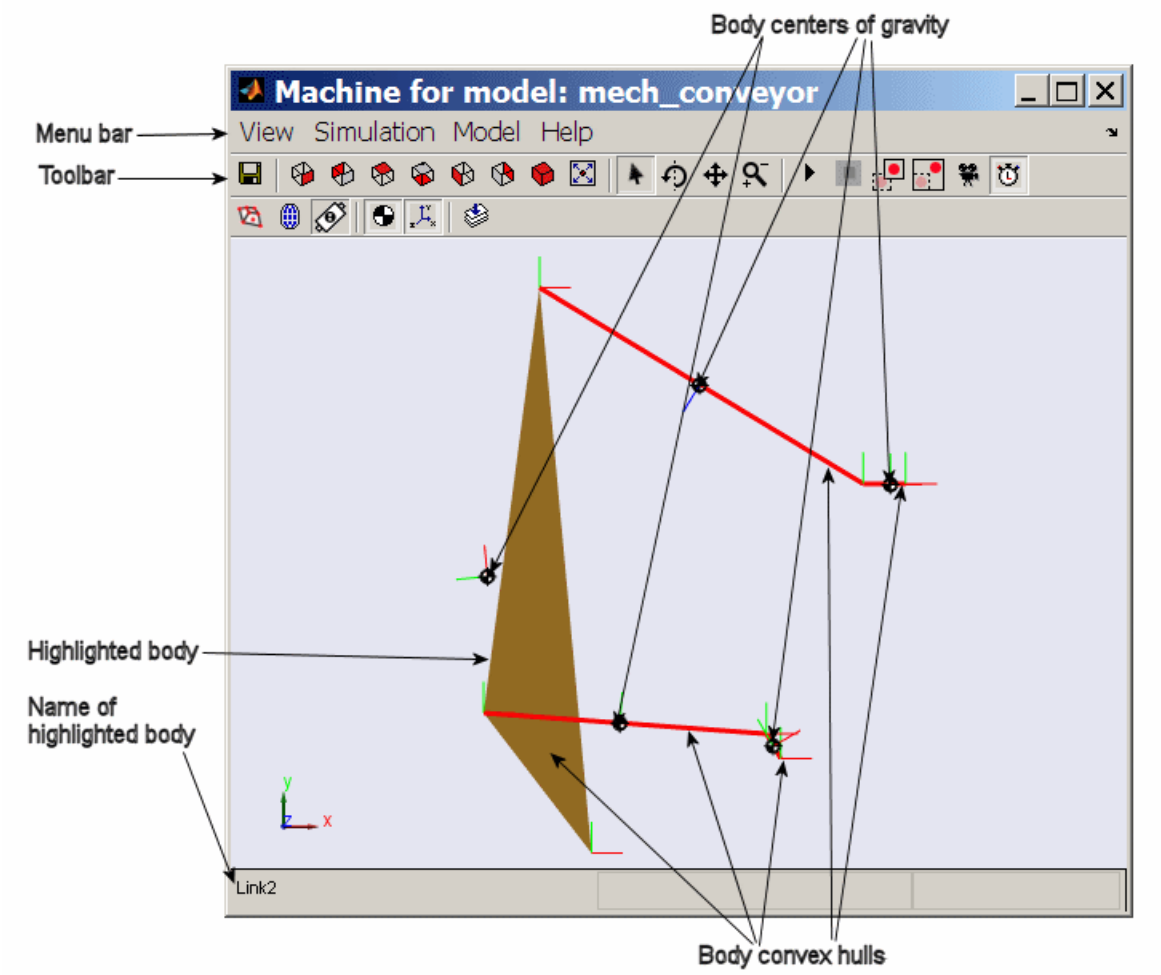
First try visualizing the conveyor with bodies displayed as convex hulls:

1. From the **Simulation** menu, select **Configuration Parameters**. The Configuration Parameters dialog box appears.
2. Select the **SimMechanics** node. In the **Visualization** area, select the **Display machines after updating diagram** and **Show animation during simulation** check boxes.
3. Leave the other defaults as they are and close the dialog. From the **Edit** menu, select **Update Diagram**.

   A SimMechanics visualization window appears, displaying the conveyor at rest in its initial state.

   The bodies are displayed in the default geometry, as convex hulls. The bodies and Body coordinate system axis triads are also displayed as defaults.
4. Change Reference Position to a nonzero value such as `0.1` or `0.2`.

5 Restart the simulation. The window animates the conveyor in motion. You can compare this motion to the plots in the scope.



6 Click a body in the visualization window. In the model window, the corresponding Body block is highlighted in red. The block name appears at the lower left of the visualization window.

7 Examine the visualization window menus and tool bar.

Here, you can reconfigure the display properties: bodies, Body CS axis triads, colored fill-in body surface patches connecting Body CSs on the same body, and viewpoint orientation.

8 Leave the visualization window open for the next set of steps.

Equivalent ellipsoids

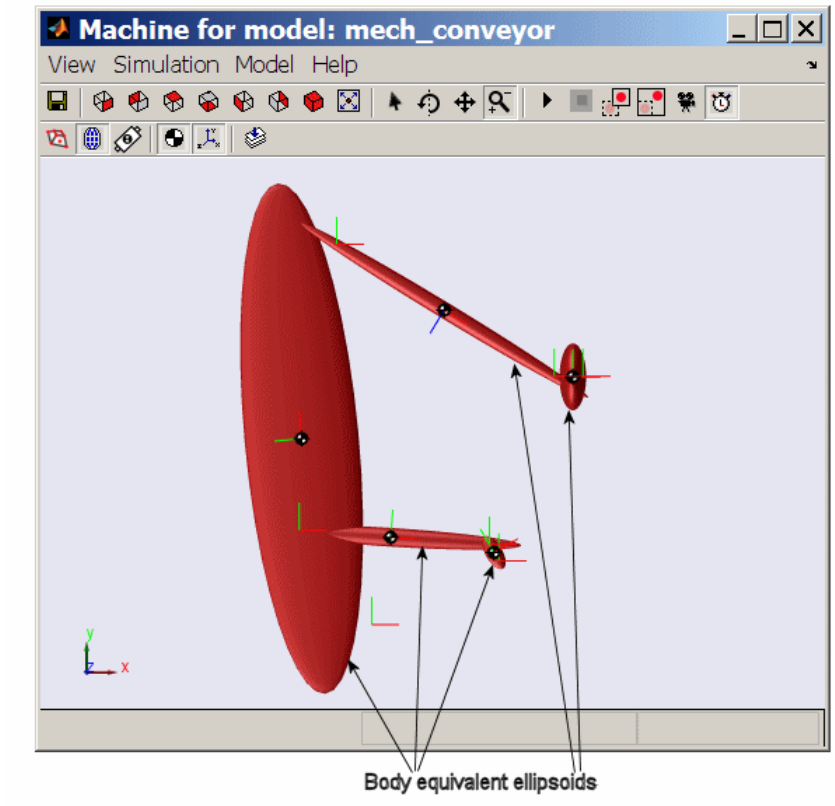
Now visualize the conveyor with bodies displayed as ellipsoids:

1 From the **Model** menu in the menu bar, select **Body Geometries > Ellipsoids**, so that a check mark appears beside the menu entry.

The display in the visualization window changes. The conveyor appears at rest in its initial state but with the bodies displayed as equivalent ellipsoids.

2 Restart the simulation. The viewer now animates the conveyor in motion.

3 Use the menus to experiment with the visualization settings. The toolbar contains most of these functions as well.



4 While the animation is running, open the Reference Position block and move the slider up and down. In addition to what you can see in the Scope plots, the window directly animates the pusher trajectory in space as the mechanism responds to your adjustment.

What You Can Do with SimMechanics Software

In this section...

About SimMechanics Software

SimMechanics software is a set of block libraries and mechanical modeling and simulation tools for use with Simulink. You connect SimMechanics blocks to normal Simulink blocks through Sensor and Actuator blocks.

The blocks in these libraries are the elements you need to model mechanical systems consisting of any number of rigid bodies, connected by joints representing translational and rotational degrees of freedom. You can represent mechanical systems with components organized into hierarchical subsystems, as in normal Simulink models. You can impose kinematic constraints, apply forces/torques, integrate the Newtonian dynamics, and measure resulting motions. See section "Simulate Conveyor Belt Model" on page 1-6.

Glossary Terms For an explanation of important terms, see the "Glossary".

Modeling Mechanical Systems

These are the major steps you follow to build and run a model representation of a machine:

1. Specify body inertial properties, degrees of freedom, and constraints, along with coordinate systems attached to bodies to measure motions and forces.

2. Set up sensors to record motions and forces, as well as actuators and force elements to initiate motions and apply forces, including continuous and discontinuous friction.

3. Start the simulation, calling the Simulink solvers to find the motions of the system, while maintaining any imposed constraints. You can also generate, compile, and run generated code versions of your models.

4. Visualize the machine while building the model and animate the simulation while running it, using the SimMechanics visualization window.

Bodies, Coordinate Systems, Joints, and Constraints

You model bodies with Body blocks specified by their masses, inertia tensors, and attached Body coordinate systems (CSs). You connect the bodies to one another with joints representing the possible motions of bodies relative to one another, the system's degrees of freedom (DoFs). You can impose kinematic constraints on the allowed relative motions of the system's bodies. These constraints restrict the DoFs or drive the DoFs as explicit functions of time.

The SimMechanics interface gives you many ways to specify CSs, constraints/drivers, and forces/torques. You can

- Attach Body CSs to different points on Body blocks to specify local axes and origins for actuating and sensing.
- Take Joint blocks from the SimMechanics library or extend the existing Joint library by constructing your own custom Joints.
- Use other Simulink tools as well as MATLAB expressions.

Defining Local Coordinate Systems

SimMechanics models automatically contain a single inertial reference frame and CS called *World*. You can also set up your own Local CSs:

- *Grounded CSs* attached to Ground blocks at rest in World but displaced from the World CS origin
- *Body CSs* fixed on and moving rigidly with the bodies

Kinematic Constraints

Specifying kinematic relations between any two bodies, you can constrain the motion of the system by connecting Constraint blocks to pairs of Bodies. Connecting Driver blocks applies time-dependent constraints.

Sensors, Actuators, Friction, and Force Elements

Sensors and Actuators are the blocks you use to interface between normal Simulink blocks and SimMechanics blocks. Force Elements represent internal forces that require no external input.

- Sensor blocks detect the motion of Bodies and Joints.
  - Sensor block outputs are Simulink signals that you can use like any other Simulink signal. You can connect a Sensor block to a Simulink Scope block and display the motions in a system.
  - You can feed these Sensor output signals back to a SimMechanics system via Actuator blocks, to specify forces/torques in the system.
- Actuator blocks specify the motions of Bodies or Joints.
  - They accept force/torque signals from Simulink and can apply forces/torques on a body or joint from these signals. The Simulink signals can include Sensor block outputs fed back from the system itself.
  - They detect discrete locking and unlocking of Joints to implement discontinuous static friction forces.
  - They specify the position, velocity, and acceleration of bodies or joints as explicit functions of time.
  - They prepare a system's initial kinematic state (positions and velocities) for the forward integration of Newtonian dynamics.

Force Elements model internal forces between bodies or acting on joints between bodies. Internal forces depend only on the positions and velocities of the bodies themselves, independent of external signals.

Simulating and Analyzing Mechanical Motion

SimMechanics software provides four modes for analyzing the mechanical systems you simulate: Forward Dynamics, Trimming, Inverse Dynamics, and

Kinematics. You can also convert any mechanical model, in any mode, to a portable, generated code version.

Mathematical Determination of Rigid Body Motion

For the forward dynamics to be mathematically solvable, the system must satisfy certain conditions:

- The masses and inertia tensors of all bodies are known.
- All forces and torques acting on each body at each instant of time are known.
- Any kinematic constraints among DoFs are specified as constraints among positions and/or velocities alone. If the constraints are mutually consistent and are fewer in number than the DoFs, the system's motion is nontrivial and can be found by integration.
- Initial conditions (initial positions and velocities) are specified and consistent with all constraints.

For inverse dynamic analysis, you specify the motions instead and obtain the forces/torques needed to produce those motions.

Forward Dynamics, Trimming, and Linearization

In the Forward Dynamics mode, a SimMechanics simulation uses the Simulink suite of ordinary differential equation (ODE) solvers to solve Newton's equations, integrating applied forces/torques and obtaining the resulting motions. The ODE solvers project the motion of the DoFs onto the mathematical manifold of the kinematic constraints and yield the forces/torques of constraint acting within the system.

Trimming. The Trimming mode allows you to use the Simulink trimming features to search for steady or equilibrium states in mechanical motion. These states, once found, are the starting point for linearization analysis.

Linearization. You can use the Simulink linearization tools to linearize the forward motion of a system and obtain its response to small perturbations in forces/torques, constraints, and initial conditions.

Inverse Dynamics

A SimMechanics simulation can solve the reverse of the forward dynamics problem, determining the forces/torques needed to produce a given set of motions that you apply to the system. Depending on the topology of your system, you choose from two SimMechanics modes for efficiently analyzing its inverse dynamics:

- The Inverse Dynamics mode works with *open* topology systems (model diagrams without closed loops).
- The Kinematics mode analyzes the motion of *closed-loop* models, including the invisible constraints imposed by loop closures.

  Constraint and Driver blocks can appear only in closed loops, so you use the Kinematics mode to analyze constraint forces/torques as well.

Tip You can use the Forward Dynamics mode to analyze inverse dynamics. But the Inverse Dynamics and Kinematics modes are optimized for such analysis and solve such problems faster and more efficiently than does Forward Dynamics.

Generating Code

SimMechanics software is compatible with Simulink Acceleration modes, Simulink Coder™, and Simulink Real-Time™ software. They let you generate code versions of the models you create originally in Simulink with block diagrams, enhancing simulation speed and model portability.

The presence of static friction in a mechanical model creates dynamical discontinuities and triggers mode iterations in Simulink. These discontinuities and mode iterations place certain restrictions on code generation.

Visualizing and Animating Models

SimMechanics software supports an internal visualization window as a powerful aid in building, animating, and debugging models. For an example of its use, see “Simulate Conveyor Belt Model” on page 1-6.

The window displays the bodies and their Body coordinate systems (CSs) in:

- Abstract, simplified shapes, convex hulls or equivalent ellipsoids. These are the standard geometries.
- Custom geometries specified by external graphics files.

You can also automatically generate SimMechanics models from a data file representing a computer-aided design (CAD) assembly exported from external CAD platforms.

Visualizing Bodies During Modeling

One way to use the visualization window is while you're building your model:

- You can open a SimMechanics visualization window before you start to build and then watch the bodies appear and be configured in the display as you create and configure them in your model window.

  This approach is especially useful if you're just starting to learn how to create complex SimMechanics models. In that case, visualization can guide you in assembling the body geometries and connections.
- You can also build a model without visualization, then open a visualization window when you have finished to see the completed model.

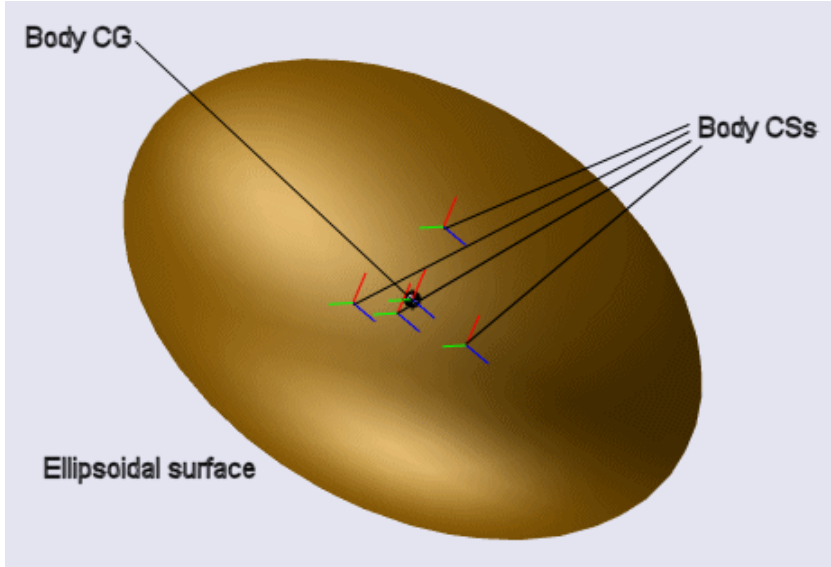
Displaying Bodies in Standard Geometries

The visualization window has two standard abstract shapes to display the bodies, one derived from body mass properties, the other from bodies' attached Body coordinate systems (CSs). These shapes are geometric schematics, based on the limited body information specified in the Body block dialog.

Mass Properties. A rigid body's dynamics are partly determined by the body's total mass and how that mass is distributed in space, as encapsulated in its inertia tensor. Any rigid body has a unique corresponding homogeneous ellipsoid with the same mass and inertia tensor.

Using these *equivalent ellipsoids* is one visualization mode of displaying a body. The relative sizes of the ellipsoid axes indicate the relative inertial moments about each axis.

Here is a rigid body displayed as an equivalent ellipsoid.

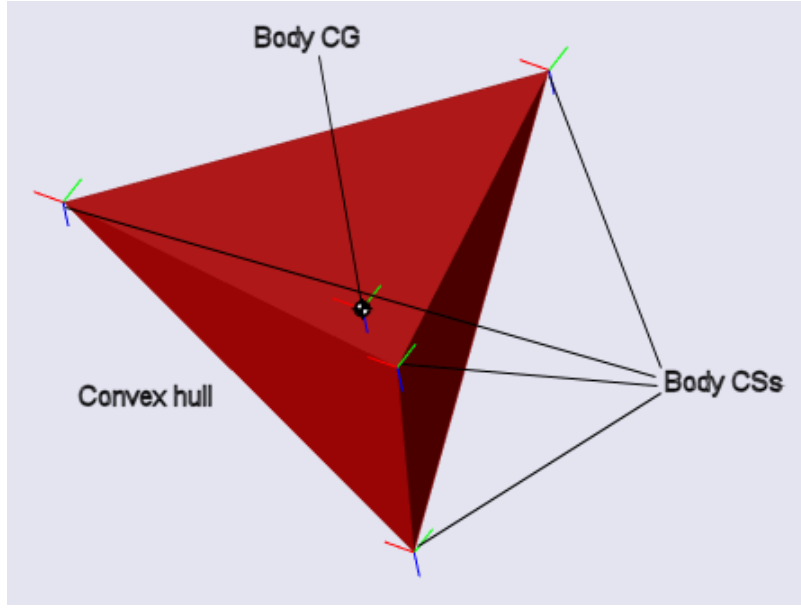


Geometric Properties. Every SimMechanics body is represented by a Body block with at least one attached Body CS. The minimum Body CS origin is located at the body's center of gravity (CG).

You can also create other Body CSs on a Body. Any Joint, Constraint/Driver, Actuator, or Sensor attached to a Body must be attached at a Body CS origin.

The set of Body CS origins can be enveloped by a surface; if there are more than three non-coplanar origins, the surface encloses a volume. The minimal surface with outward-bending curvature enveloping this set is the *convex hull*, which is the other abstract shape available for visualizing a body in space. Fewer than four CS origins produce simpler Body figures. The convex hull excludes the Body CG CS.

Here is the same body as a convex hull. The four Body CS origins are non-coplanar in this case, and the hull is a tetrahedron.



Animating Motion During Simulation

Besides displaying your model's bodies either while you build the model or as a completed model, you can also keep the visualization window open while a model is running in the Simulink model window. The window animates the simulation of the bodies' motions, whether you choose to display the bodies as ellipsoids or as convex hulls, and moves in parallel with model changes on the Simulink side.

Interfacing with Computer-Aided Design

You can use the SimMechanics importer to automatically generate a SimMechanics model based on an external data file previously exported from a supported CAD platform. This data file captures the dynamically important features of a CAD assembly representing a mechanical system. The resulting model, once generated, can be modified and expanded like any other SimMechanics model.

2

Modeling, Simulating, and Visualizing Simple Machines

Constructing simple SimMechanics models is easy to learn if you already know how to make Simulink models. If you are not already familiar with Simulink, see the Simulink documentation.

Introducing the SimMechanics Block Libraries

In this section...

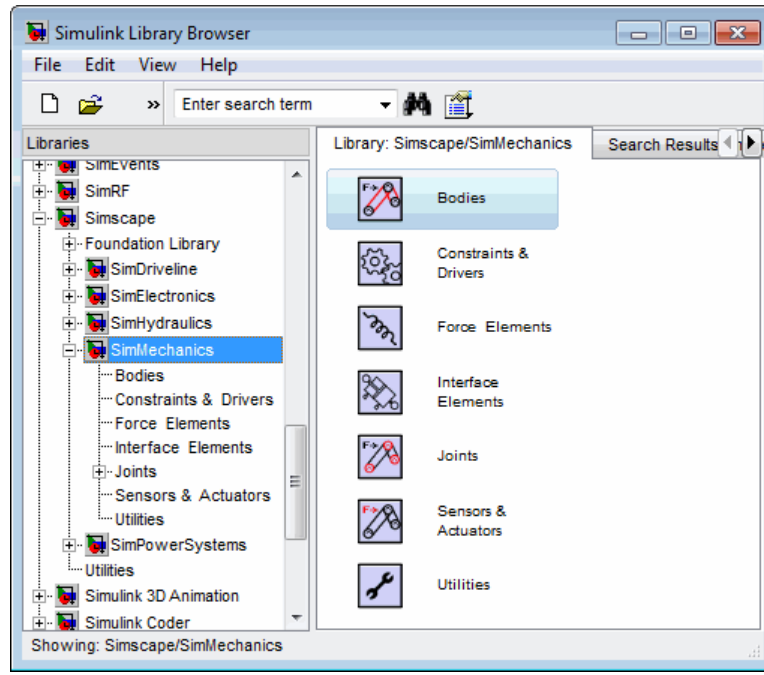
About the SimMechanics Block Library

SimMechanics software is organized into a set of libraries of closely related blocks. This section shows how to view these libraries and gives you a summary of what they contain.

Accessing the Libraries

There are several ways to open the SimMechanics block library.

You can access the blocks through the Simulink Library Browser. Open the browser by clicking the Simulink button  . Expand the Simscape entry, then the SimMechanics subentry, in the contents tree.

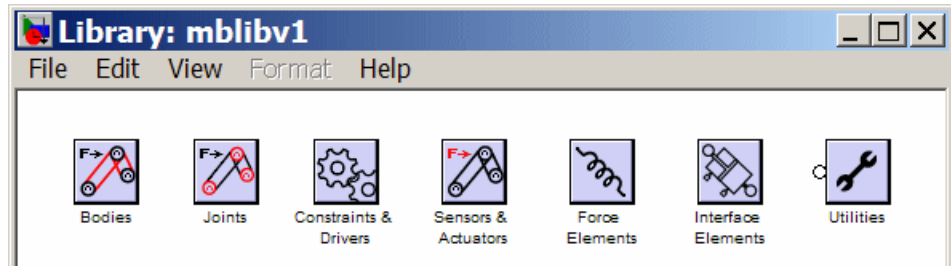


You can also access the blocks directly inside the SimMechanics library in several ways:

- In the Simulink Library Browser, right-click the SimMechanics subentry under Simscape and select **Open SimMechanics Library**. The library appears.
- Click the **Start** button in the lower left corner of your MATLAB desktop. In the pop-up menu, select **Simulink**, then **SimMechanics**, then **Block Library**.
- Enter `mechlib` at the MATLAB command line.

SimMechanics Library

Once you perform one of these steps, the SimMechanics library opens.



Tip This library displays the top-level block groups. You can expand each library by double-clicking its icon. The Joints library contains second-level sublibraries.

The next section summarizes the blocks of each library and their use.

Using the Libraries

The SimMechanics block library is organized into separate libraries, each with a different type of mechanical block.

Bodies

The Bodies library provides the Body block for representing bodies defined by their mass properties (masses and inertia tensors), their positions and orientations, and their attached Body coordinate systems (CSs). This library also contains the Ground block representing immobile ground points, which have their own Grounded CSs, and the Machine Environment block, for configuring the mechanical settings of a SimMechanics block diagram.

Joints

The Joints library provides blocks to represent the relative motions between bodies as degrees of freedom (DoFs). The library is made up of assembled Joints listed individually and two sublibraries of specialized Joint blocks.

An assembled joint restricts the Body CSs on the two bodies to which it is connected. The assembled Joints are the primitive Prismatic, Revolute, and Spherical blocks and ready-made composite Joints. Unless it is explicitly labeled as disassembled, you can assume a generic Joint block is assembled.

Joints/Disassembled Joints. The Disassembled Joints sublibrary provides blocks for disassembled joints, modified joints that do not restrict the Body CSs on the two connected bodies or the DoF axes of the two bodies. You can only use Disassembled Joints to close a loop in your machine. You cannot sense or actuate Disassembled Joints.

Joints/Massless Connectors. The Massless Connectors sublibrary provides blocks for massless connectors, composite joints whose DoFs are separated by a fixed distance. You cannot actuate or sense Massless Connectors.

Constraints & Drivers

The Constraints & Drivers library provides blocks to specify prior restrictions on DoFs between Bodies. These restrictions can be time-independent constraints or time-dependent driving of DoFs with Simulink signals.

Sensors & Actuators

The Sensors & Actuators library provides blocks for sensing and initiating the motions of joints and bodies. These blocks play a special role in connecting SimMechanics blocks to other Simulink blocks, as described in “Connecting SimMechanics Blocks”, “Applying Motions and Forces”, and “Sensing Motions and Forces”.

Force Elements

The Force Elements library provides blocks for creating forces or torques between bodies. These blocks model forces internal to your machine.

Interface Elements

The Interface Elements libraries includes blocks to connect three-dimensional machines modeled with SimMechanics blocks and one-dimensional mechanical Simscape circuits.

Utilities

The Utilities library contains miscellaneous blocks useful in building models.

Build and Run a Mechanical Model

In this section...

About Machine Modeling and Simulation

To become comfortable building mechanical models, you might find it helpful to work through the guided examples in subsequent sections of how to configure and put together elements of SimMechanics software to simulate simple machines. This section gives you an overview of the model-building process before you start:

- "Essential Steps to Build a Model" on page 2-7
- "Essential Steps to Configure and Run a Model" on page 2-9

The special terms used in this guide are summarized in the "Glossary".

Essential Steps to Build a Model

You use the same basic procedure for building a SimMechanics model regardless of its complexity. The steps are similar to those for building a regular Simulink model. More complex models add steps without changing these basics.

1 *Select Ground, Body, and Joint blocks.* From the Bodies and Joints libraries, drag and drop the Body and Joint blocks needed to represent your machine, including a Machine Environment block and at least one Ground block, into a Simulink model window.

   The Machine Environment block represents your machine's mechanical settings.

   Ground blocks represent immobile ground points at rest in absolute (inertial) space.

Body blocks represent rigid bodies.

Joint blocks represent relative motions between the Body blocks to which they are connected.

2 *Position and connect blocks.* Place Joint and Body blocks in proper relative position in the model window and connect them in the proper order. The essential result of this step is creation of a *valid tree* block diagram made of

Machine Env — Ground — Joint — Body — Joint — Body — ... — Body

with an open or closed topology and where at least one of the bodies is a Ground block. Connect exactly one environment block to a Ground.

A Body can have more than two Joints attached, marking a branching of the sequence. But Joints must be attached to two and only two Bodies.

3 *Configure Body blocks.* Click the Body blocks to open their dialog boxes; specify their mass properties (masses and moments of inertia), then position and orient the Bodies and Grounds relative to the World coordinate system (CS) or to other CSs. You set up Body CSs here.

4 *Configure Joint blocks.* Click each of the Joint blocks to open its dialog box and set translation and rotation axes and spherical pivot points.

5 *Select, connect, and configure Constraint and Driver blocks.* From the Constraints & Drivers library, drag, drop, and connect Constraint and Driver blocks in between pairs of Body blocks. Open and configure each Constraint/Driver's dialog box to restrict or drive the relative motion between the two respective bodies of each constrained/driven pair.

6 *Select, connect, and configure Actuator and Sensor blocks.* From the Sensors & Actuators library, drag and drop the Actuator and Sensor blocks that you need to impart and sense motion. Reconfigure Body, Joint, and Constraint/Driver blocks to accept Sensor and Actuator connections. Connect Sensor and Actuator blocks. Specify control signals (applied forces/torques or motions) through Actuators and measure motions through Sensors.

Actuator and Sensor blocks connect SimMechanics blocks to normal Simulink blocks. You cannot connect SimMechanics blocks to regular Simulink blocks otherwise. Actuator blocks take inport signals from

normal Simulink blocks (for example, from the Simulink Sources library) to actuate motion. Sensor block output ports generate Simulink signals that you can feed to normal Simulink blocks (for example, from the Simulink Sinks library).

In the most straightforward case, you apply forces/torques and initial conditions, then start the simulation in the Forward Dynamics mode to obtain the resulting motions. In the Kinematics and Inverse Dynamics modes, you apply motions to all independent degrees of freedom. With these modes, you can find the forces/torques needed to produce these imposed motions.

7 *Encapsulate subsystems.* Systems made from SimMechanics blocks can function as subsystems of larger models, like subsystems in normal Simulink models. You can connect an entire SimMechanics model as a subsystem to a larger model by using the Connection Port block in the Utilities library.

Essential Steps to Configure and Run a Model

After you've built your model as a connected block diagram, you need to decide how you want to run your model, configure SimMechanics and Simulink settings, and set up visualization.

- You can choose from among four SimMechanics analysis modes for simulating a machine. The mode you will probably use most often is Forward Dynamics.

  But a more complete analysis of a machine makes use of the Kinematics, Inverse Dynamics, and Trimming modes as well. You can create multiple versions of the model, each with the same underlying machine, but connected to Sensors and Actuators and configured differently for different modes.

- You can also use the powerful SimMechanics visualization and animation features. You can visualize your model as you build it or after you are finished but before you start the simulation, as a tool for debugging the model's geometry. You can also animate the model as you simulate.

- Choose the analysis mode, as well as other important mechanical settings, in your Machine Environment dialog. Start visualization and adjust

Simulink settings in the Simulink Configuration Parameters dialog. See "Model and Simulate a Closed-Loop Machine" on page 2-38 for an example.

The tutorials of this chapter introduce you to most of these steps. The first, in the next section, shows you how to configure the most basic blocks in any model: Machine Environment, Ground, Body, and a Joint, in order to create a simple pendulum model. The second tutorial explains how to visualize and animate the pendulum.

Caution You might want to make modifications to these tutorial models. To avoid errors,

- Do not attempt to connect Simulink signal lines directly to SimMechanics blocks other than Actuators and Sensors
- Keep the collocation of the Body coordinate system origins on either side of each assembled Joint to within assembly tolerances

  You should save multiple versions of models as you try different analysis modes and configurations.

Model and Simulate a Simple Machine

In this section...

Model a Simple Pendulum

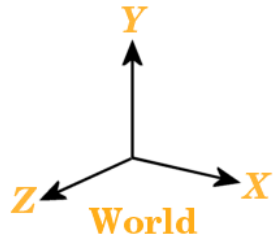
In this first tutorial, you drag, drop, and configure the most basic blocks needed for any mechanical model, as well as add some sensors to measure motion. The tutorial guides you through the most basic aspects of model-building.

The end result is a model of a simple pendulum, a system with one body and open topology. The pendulum is a swinging steel rod. We strongly recommend that you work through this tutorial first before moving on to, "Visualize a Simple Machine" on page 2-32.

A Simple Pendulum: A Swinging Steel Rod

The World Coordinate System and Gravity

Before you configure a Ground block, you need to understand the internally defined fixed or "absolute" SimMechanics coordinate system (CS) called *World*. The World CS sits at rest in the inertial reference frame also called World. The World CS has an origin (0,0,0) and a triad of right-handed, orthogonal coordinate axes.



The default World coordinate axes are defined so that

+*x* points right

+*y* points up (gravity in -*y* direction)

+*z* points out of the screen, in three dimensions

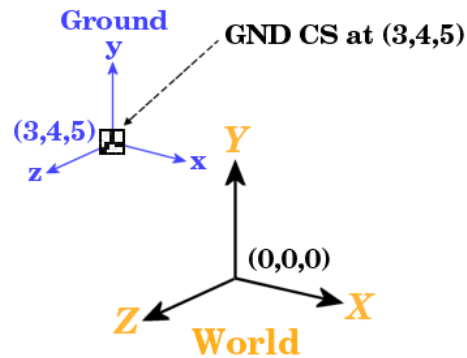
The vertical direction or up-and-down is determined by the gravity vector direction (acceleration ***g***) relative to the World axes. Gravity is a background property of a model that you can reset before starting a simulation, but does not dynamically change during a simulation.

See "Configuring SimMechanics Models in Simulink" for displaying global mechanical properties of SimMechanics models.

Set Up the Ground

World serves as the single absolute CS that defines all other CSs. But you can create additional *ground points* at rest in World, at positions other than the World origin, by using Ground blocks. Ground blocks, representing ground points, play a dynamical role in mechanical models. They function as immobile bodies and also serve to implement a machine's mechanical environment.

Minimum Ground Blocks Every machine model must have *at least one* Ground block. Exactly one Ground block in every machine must be connected to a Machine Environment block.

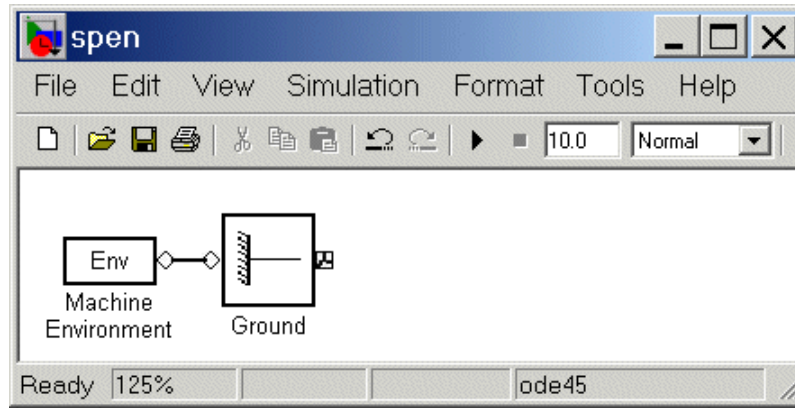


A Ground Point Relative to World

Steps to Set Up the Ground Block

Now place a fixed ground point at position (3,4,5) in the World CS:

1. In the SimMechanics library, open the Bodies library.
2. Drag and drop a Ground and a Machine Environment block from the Bodies library into the model window.
3. Open the Ground block dialog box. Into the **Location [x y z]** field, enter the vector `[3 4 5]`. Select the **Show Machine Environment port** check box. Click **OK** to close the dialog. Connect the environment block.



Ground Properties

At every ground point, a *Grounded CS* is automatically created:

- The origin of each Grounded CS is the ground point itself.
- The Grounded CS axes are always fixed to be parallel to the World CS axes, as shown in the figure A Ground Point Relative to World on page 2-13.

Configure the Body

While you need one Machine Environment and at least one Ground block to make a mechanical model, a real machine consists of one or more rigid bodies. So you need to translate the components of a real machine into block

representations. This section explains how you use a Body block to represent each rigid body in your system:

Although the body is the most complicated component of a machine, the Body block does not use the full geometric shape and mass distribution of the body. A SimMechanics model uses only certain mass properties and simplified geometric information about the body's center of gravity, its orientation, and the coordinate systems attached to the body.

Setting these properties sets the body's initial conditions of motion, if you do nothing else to the Body block or its connected Joints before simulating.

Characteristics of a Body Block

The main characteristics of a Body block are its *mass properties*, its *position* and *orientation* in space, and its attached *Body coordinate systems* (CSs).

The mass properties include the *mass* and *inertia tensor*. The mass is a real, positive scalar. The inertia tensor is a real, symmetric 3-by-3 matrix. It does not have to be diagonal.

The position of the body's *center of gravity* (CG) and *orientation* relative to some coordinate system axes indicate where the body is and how it is rotated. These are the body's initial conditions during construction of the model and remain so when you start the simulation, unless you change them before starting.

The attached *Body CSs* (their origins and coordinate axes) are fixed rigidly in the body and move with it. The minimum CS set is one, the CS at the CG (the CG CS), with its CS origin at the center of gravity of the body. The default CS set is three, the CG CS and two additional CSs called CS1 and CS2 for connecting to Joints on either side. See the next section, "Configure the Joint" on page 2-21.

Beyond the minimum CS at the CG, you can attach as many Body CSs on one Body as you want. You need a separate CS for each connected Joint, Constraint, or Driver and for each attached Actuator and Sensor.

The inertia tensor components are interpreted in the CG CS, setting the orientation of the body relative to the CG CS axes. The orientation of the CG CS axes relative to the World axes then determines the absolute initial orientation of the body. Since the CG CS axes remain rigidly fixed in the body during the simulation, this relative orientation of the CG CS axes and the body does not change during motion. The inertia tensor components in the CG CS also do not change. As the body rotates in inertial space, however, the CG CS axes rotate with it, measured with respect to the absolute World axes.

Properties of the Simple Pendulum Body

The simple pendulum is a uniform, cylindrical steel rod of length 1 meter and diameter 2 cm. The initial condition is the rod lying parallel to the negative x-axis, horizontal in the gravity field. One end of the rod, the fixed pivot for the rod to swing, is located at the ground point (3,4,5). Its coordinate system is called CS1. The center of gravity and the origin of the CG CS is the geometric center of the rod. Take the CG CS axes to be parallel to the World axes as you set up the pendulum.

Uniform steel has density $\rho = 7.93$ gm/cc (grams per cubic centimeter). In the CG CS here, the inertia tensor $\boldsymbol{I}$ is diagonal, and I_{zz} controls the swinging about the z-axis, in the x-y plane. The inertia tensor is always evaluated with the origin of coordinates at the CG. For a rod of length $L = 1$ m and radius $r = 1$ cm, the mass $m = \rho\pi r^2 L = 2490$ gm (grams), and the inertia tensor $\boldsymbol{I}$ reads

$$\begin{pmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{pmatrix} = \begin{pmatrix} \frac{mr^2}{2} & 0 & 0 \\ 0 & \frac{mL^2}{12} & 0 \\ 0 & 0 & \frac{mL^2}{12} \end{pmatrix} = \begin{pmatrix} 1250 & 0 & 0 \\ 0 & 2.08\times10^6 & 0 \\ 0 & 0 & 2.08\times10^6 \end{pmatrix}$$

in gm-cm^2 (gram-centimeters^2). The x-axis is the cylinder's symmetry axis. Thus $I_{yy} = I_{zz}$.

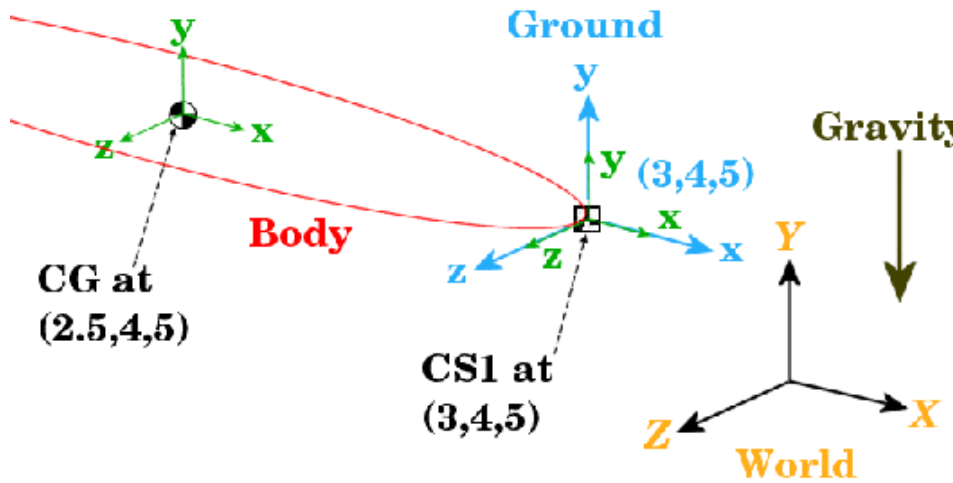
The mass and geometric properties of the body are summarized in the following table and depicted in the figure Equivalent Ellipsoid of Simple Pendulum with Body Coordinate Systems on page 2-17.

Body Data for the Simple Pendulum

Property	Value
Mass (gm)	2490
Inertia tensor (kg-m^2)	[1.25e-4 0 0; 0 0.208 0; 0 0 0.208]
CG Position/Origin (m)	[2.5 4 5]
CS1 Origin (m)	[3 4 5]

Configuring the Body Dialog

Take the steps to configuring a Body block in several stages.



Equivalent Ellipsoid of Simple Pendulum with Body Coordinate Systems

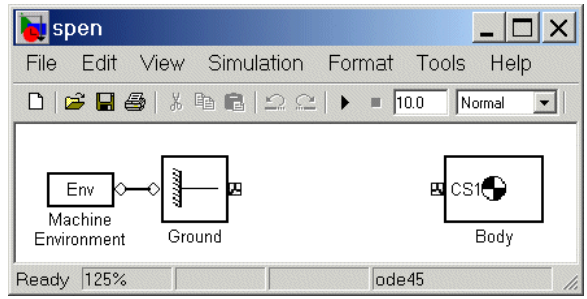
Add the Body Block. To start working with the Body block:

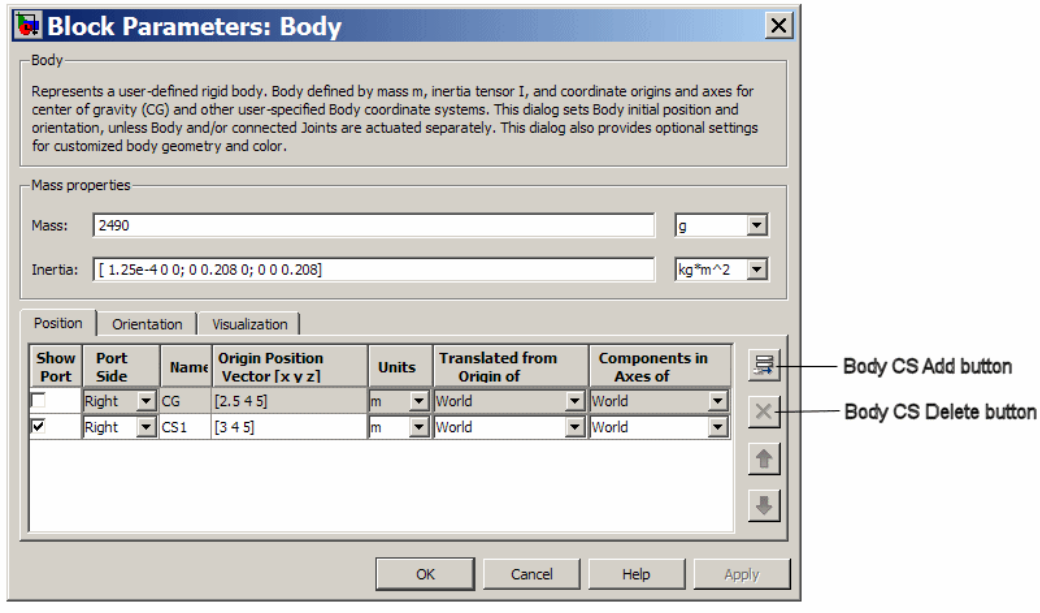
1 Open the Bodies library in the SimMechanics library.

2 Drag and drop a Body block into your model window.

3 Open the Body block dialog box. Note the two main areas you need to configure:

- **Mass properties** — These are the mass and inertia tensor.
- Body coordinate systems — These are the details about the position and orientation of the Body CSs.





Specify Mass Properties. Now enter the body's mass and inertia tensor:

1. Use the data from the table Body Data for the Simple Pendulum on page 2-17.

   In the **Mass** field, enter `2490` and change the units to `g` (grams).

2. In the **Inertia** field, enter `[1.25e-4 0 0; 0 0.208 0; 0 0 0.208]` and leave the default units as `kg-m`^{2}.

Specify Body Coordinate Systems (Position). Enter the translational position of the body and its Body CS origins in space:

1. Use the data from the table Body Data for the Simple Pendulum on page 2-17, and work on the **Position** tab. Vectors are assumed translated from the World origin and oriented to the World axes.

2 Note the three default CSs in the Body dialog box. The CS at the CG is necessary for any Body, and you will connect CS1 to the Ground with a Joint shortly.

   Delete CS2 by selecting its line in the Body CS list and clicking the **Delete** button in the Body CS controls.

   You have two already existing CSs not on this Body that you can use to specify the positions of the Body CS origins that are on this Body:

   - Preexisting World origin at `[0 0 0]`
   - The Adjoining CS on the neighboring body, in this case the Grounded CS origin at `[3 4 5]`

3 Specify the CG and CS1 origins relative to World:

   a In the pull-down menu under **Translated from Origin of**, choose `World` for both coordinate systems, CG and CS1.

   b Under **Origin Position Vector**, specify the position of the origin of each CS, translated from the World origin:

      `[3 4 5]` for CS1

      `[2.5 4 5]` for CG

4 Select a CS relative to whose coordinate axes the components of the vectors in the last step are measured. You choose these CS axes in the **Components in Axes of** menu. Select `World` for both CSs. Leave the units as `m` (meters).

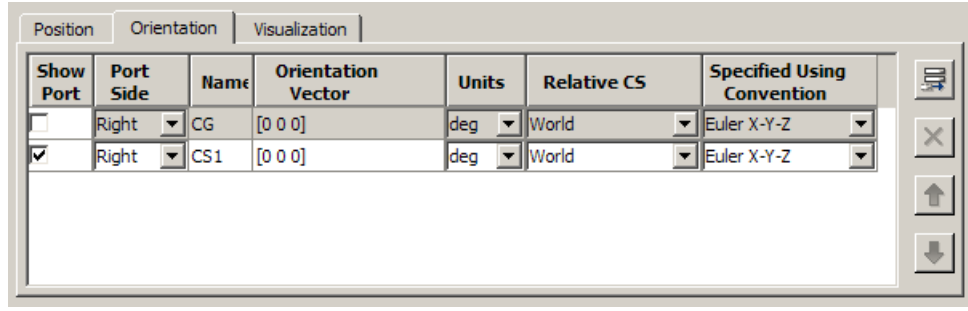
Specify Body Coordinate Systems (Orientation). Enter the rotational orientation of the body and its Body CS axes in space:

1 Work on the **Orientation** tab. The default orientation for all CS axes is parallel to World. The sign of all rotations is determined by the *right-hand rule*.

   In the figure Equivalent Ellipsoid of Simple Pendulum with Body Coordinate Systems on page 2-17, the CS1 and CG axes are oriented parallel to the World axes, so the CS1 and CG axes need no rotation.

2 For both CSs, set the **Relative to CS** menu to `World`.

3 For CG and CS1, leave the **Orientation Vector** at default `[0 0 0]` and the **Specified Using Convention** at default `Euler X-Y-Z`. Close the Body dialog.



Configure the Joint

A mechanical system is made up of Bodies with geometric and mass information. But Bodies carry no information of how they move. The possible directions of motion that a Body can take are called its *degrees of freedom* (DoFs), and this section explains how you represent these DoFs by Joint blocks:

- "How to Connect a Joint Between Two Bodies" on page 2-22
- "Choosing a Revolute Joint for the Simple Pendulum" on page 2-22

DoFs Are Relative SimMechanics DoFs and the Joints that represent them are *relative* DoFs. That is, DoFs represent the possible motions between one body and another. So a DoF is defined by a pair of bodies, and you must connect every Joint to two and only two Bodies.

One (but not both) of the members of such a pair of Bodies can be a Ground. The other member Body of such a pair then has its motion defined relative to a fixed ground point. This fixed ground point does not have to be the same as the World origin. A system can have many such Ground-Body pairs and *must have at least one.*

How to Connect a Joint Between Two Bodies

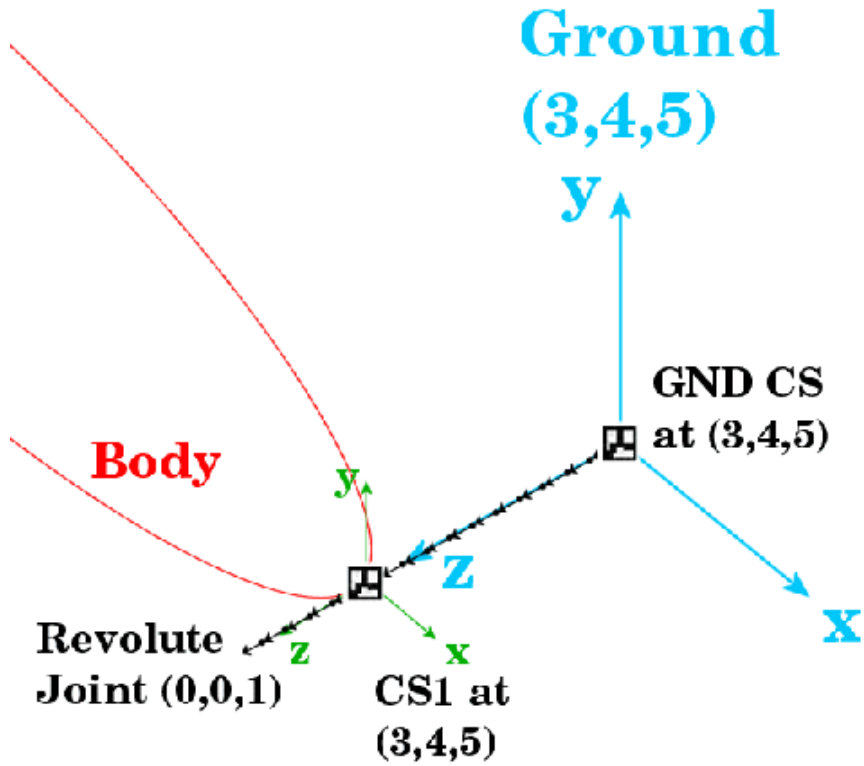
You represent relative motion of bodies with respect to one another by connecting their Body blocks with Joints. You can connect a Body to one or more Joints.

A Joint block is always connected to a specific point on the Body on either side of the Joint. The specific point for anchoring a Joint on a Body is the origin of a Body CS, and a Joint is therefore connected on one side to one Body at a Body CS origin, and on the other side to the other Body at a Body CS origin.

Usually a Body is connected to a Joint on either side, so the default you saw earlier in this tutorial for Body CSs in the Body dialog box is three Body CSs: the CS at the center of gravity (CG) and two other CSs (CS1 and CS2). But a Body at the end of a Body — Joint — ... — Body chain is connected to only one Joint.

Choosing a Revolute Joint for the Simple Pendulum

In spite of the complexity of the concepts implicit in a Joint, the actual configuration of a Joint block is fairly simple. Here you insert and configure one revolute Joint block between the Ground and Body blocks you've already put into the model window.



A Simple Pendulum Connected to Ground by a Revolute

Configure the Revolute Joint Block. The geometry of the Joint connection is shown in the figure preceding. The ground point at (3,4,5) and the CS1 at (3,4,5) are actually the same point in space, but have been separated in the figure for clarity. The revolute rotation axis is along the +*z* direction:

1. Open the Joints library in the block library.
2. Drag and drop a Revolute block into your model window.
3. Rotate the Revolute block so that you can connect the base (B) side of the Joint to the Ground block and the follower (F) side of the Joint to the Body block. Make the two connections.

4 Open the Revolute dialog box. In the **Parameters** area, on the **Axes** tab, configure the rotation axis to the World *z*-axis:

   a Enter `[0 0 1]` under **Axis of Action [x y z]**.

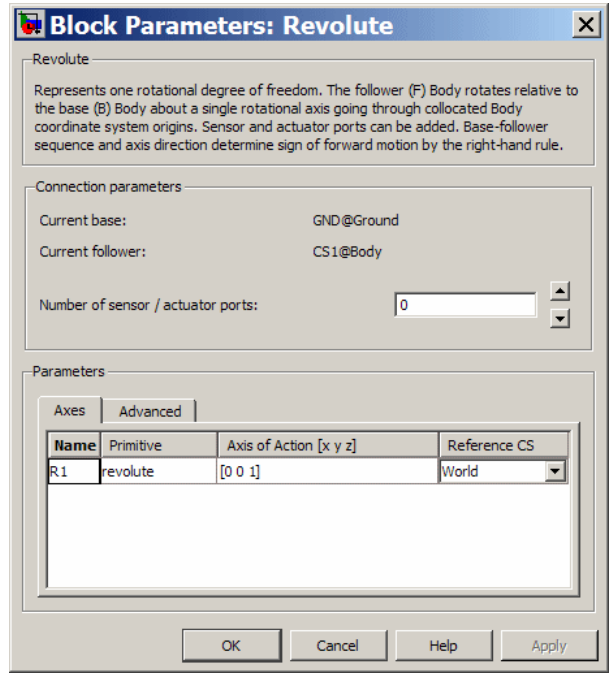
   b Leave the **Reference CS** at `World`.

   c Ignore the **Advanced** tab.

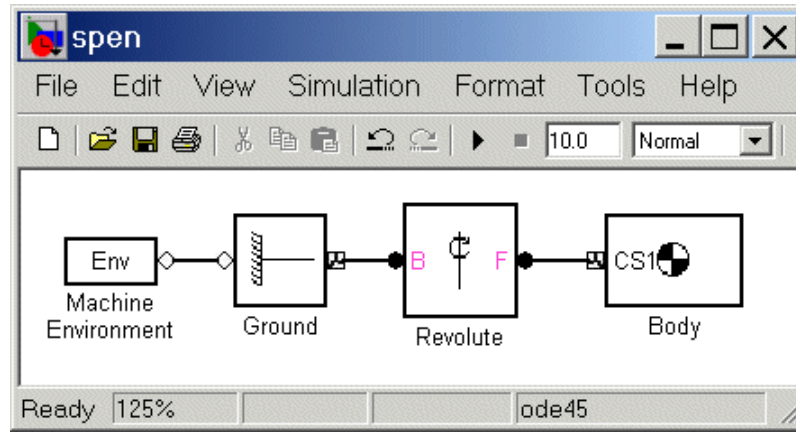
   Note several important things:

   - Under **Connection parameters**, the **Current base** is located at `GND@Ground`, which is the Grounded CS associated with the Ground block located at (3,4,5) in World.
   - Under **Connection parameters**, the **Current follower** is located at `CS1@Body`, which is the CS1 on Body1 located at (3,4,5) in World.
   - This Joint's directionality runs from Ground to Body along the +*z*-axis.

5 Close the Revolute dialog box.



Congratulations — you have now finished the simplest possible model of a machine: a connected block diagram of Ground–Joint–Body. Your model window should look like this.



Add Sensor and Run Simulation

To measure the motion of the pendulum as it swings, you need to connect one or more Simulink Scope blocks to your model. The SimMechanics library of Actuators and Sensor blocks gives you the means to input and output Simulink signals to and from SimMechanics models. Sensors allow you to watch the mechanical motion once you start the simulation, as the following explain:

- "Connect and Configure Sensor" on page 2-25
- "Set Up Machine Environment and Configuration Parameters" on page 2-27
- "Run Simulation" on page 2-29

Connect and Configure Sensor

In this example, you measure the angular motion of the revolute joint:

1 In the block library, open the Sensors & Actuators library. Drag and drop a Joint Sensor block into your model window.

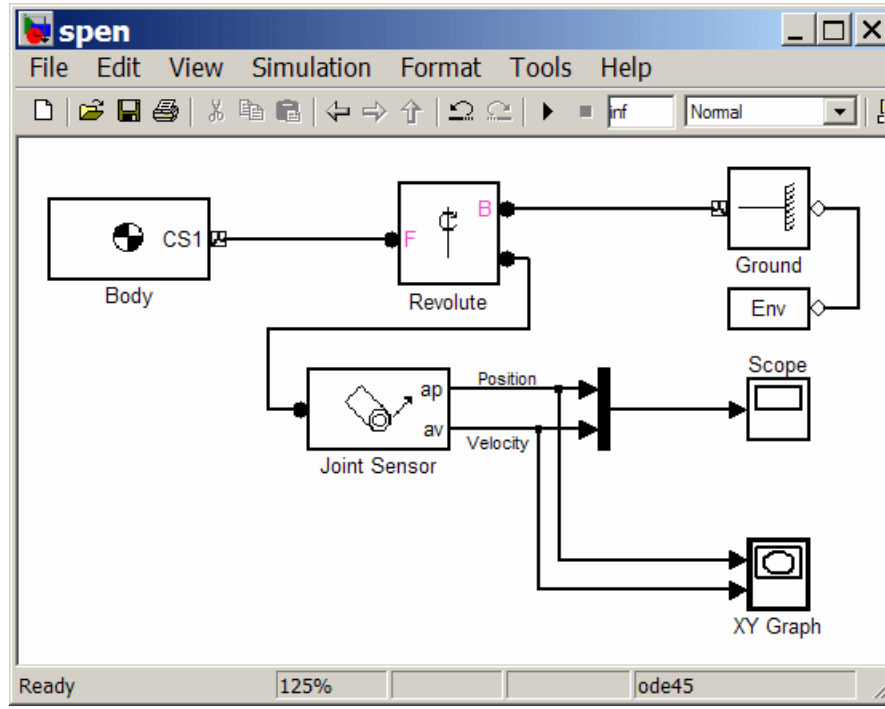
2 Open the Revolute block. Change **Number of sensor/actuator ports** from 0 to 1 using the spinner menu. An open connector port ○ appears on the side of Revolute. Close Revolute.

3 Connect this connector port to the connector port on the Joint Sensor block. The open connector port changes to solid ●.

4 Open Joint Sensor. Select the **Angle** and the **Angular velocity** check boxes. Unselect the **Output selected parameters as one signal** check box.

Leave the other defaults. Close the Joint Sensor block.

5 Open the Simulink Library Browser. From the Sinks library, drag and drop a Scope block and an XY Graph block into your model window. From the Signal Routing library, drag and drop a Mux block as well. Connect the Simulink outports > on the Joint Sensor block to the Scope and XY Graph blocks as shown.



The lines from the outports > to the Scope and XY Graph blocks are normal Simulink signal lines and can be branched. You cannot branch the lines connecting SimMechanics blocks to each another at the round connector ports ●.

6 Save your model for future reference as `spen.mdl`.

You now need to configure the global parameters of your model before you can run it.

Set Up Machine Environment and Configuration Parameters

The Configuration Parameters dialog box is a standard feature of Simulink. Reset its entries for this model to obtain more accurate simulation results.

1 In the Simulink menu bar, open the **Simulation** menu and click **Configuration Parameters** to open the Configuration Parameters dialog.

2 Select the **Solver** node of the dialog. Under **Solver options**, change **Relative tolerance** to `1e-6` and **Absolute tolerance** to `1e-4`. Change **Max step size** to `0.2`.

   If you want the simulation to continue running without stopping, change **Stop time** to `inf`. The pendulum period is approximately 1.6 seconds.

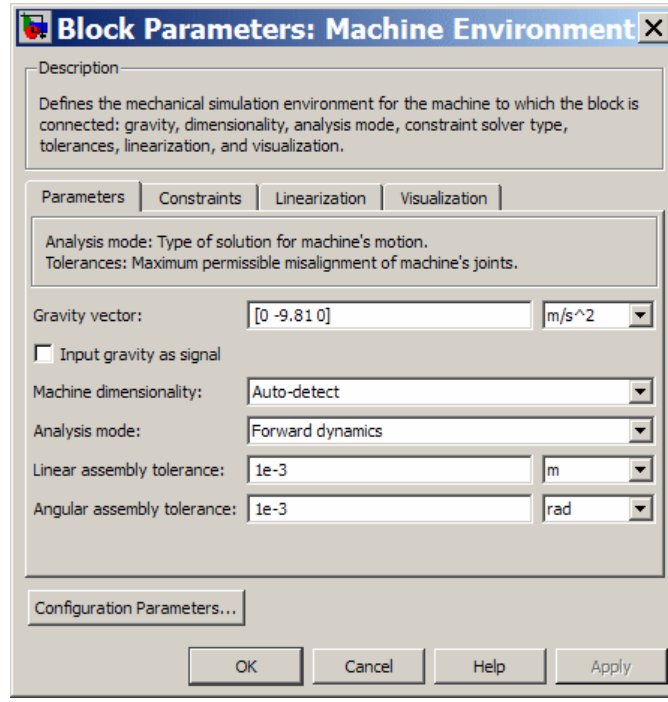
3 Close the Configuration Parameters dialog box.

A special feature of SimMechanics models is the Machine Environment block.

1 Open your block diagram's Machine Environment block dialog.

   Note the default **Gravity vector**, `[0 -9.81 0] m/s`^{2}, which points in the *-y* direction, as shown in the figure Equivalent Ellipsoid of Simple Pendulum with Body Coordinate Systems on page 2-17. The gravitational acceleration g = 9.81 m/s^2.

2 Close the Machine Environment dialog.



Run Simulation

You can now start your simulation and watch the pendulum motion via the Scope and XY Graph blocks:

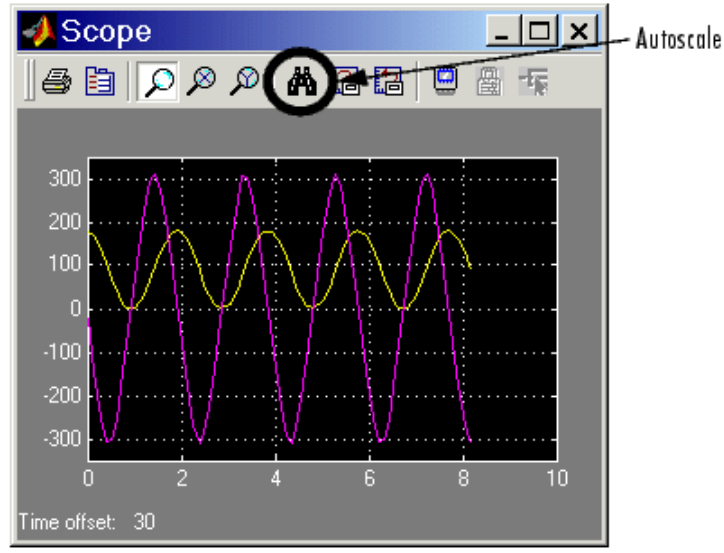
1 Open the XY Graph block dialog box. Set the following parameters.

Parameter	Value
`x-min`	0
`x-max`	200
`y-min`	-500
`y-max`	500

Leave **Sample time** at default and close the dialog.

2. Open the Scope block and start the model. The XY Graph opens automatically when you start the simulation.

3. View the full motion of both angle and angular velocity (in degrees and degrees per second, respectively) as functions of time in Scope. Click **Autoscale** if the motion is not fully visible.

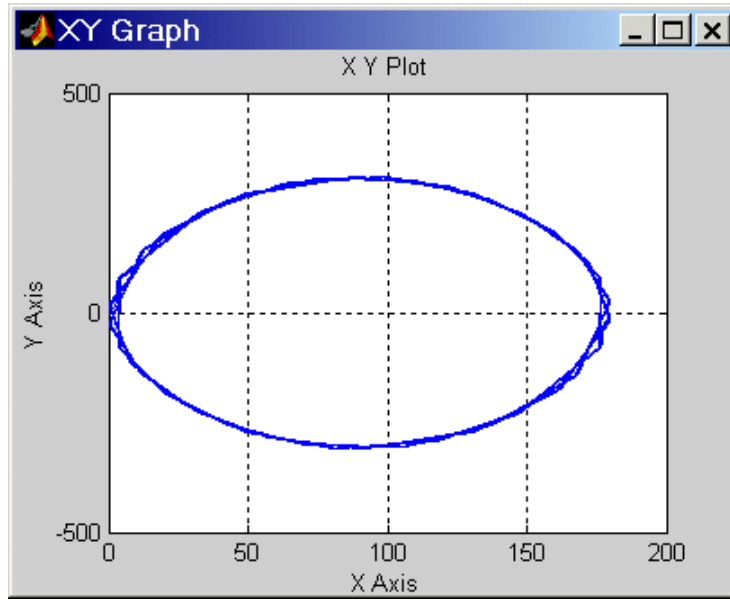


The motion is periodic but not simple harmonic (sinusoidal), because the amplitude of the swing is so large (180 degrees from one turning point to the other). Note that the zero of angle is the initial horizontal angle, not the vertical. The zeros of motion are always the initial conditions.

The XY Graph shows the angle versus angular velocity, with no explicit time axis. These two variables trace out a figure similar to an ellipse, because of the conservation of total energy E:

$$\frac{1}{2}J\left(\frac{d\theta}{dt}\right)^2 + mgh*(1-\sin\theta) = E = \text{constant}$$

where $J = I_{zz} + mL^2/4$ is the inertial moment of the rod about its pivot point (not the center of gravity). The two terms on the left side of this equation are the kinetic and potential energies, respectively. The coordinate-velocity space is the *phase space* of the system.



Phase Space Plot of Simple Pendulum Motion: Angular Velocity Versus Angle

The directionality of the Revolute Joint assumes that the rotation axis lies in the +*z* direction. Looking at the pendulum from the front, follow the figures

- A Ground Point Relative to World on page 2-13
- Equivalent Ellipsoid of Simple Pendulum with Body Coordinate Systems on page 2-17
- A Simple Pendulum Connected to Ground by a Revolute on page 2-23

Positive angular motion from this perspective is counterclockwise, following the right-hand rule.

The next tutorial walks you through visualizing and animating this same simple pendulum model.

Visualize a Simple Machine

In this section...

Visualizing and Animating the Simple Pendulum

In this section, you learn how to view the swinging steel rod of the model introduced in the last section using the SimMechanics visualization window. Use your saved `spen.mdl` model, or use the mech_spen example model.

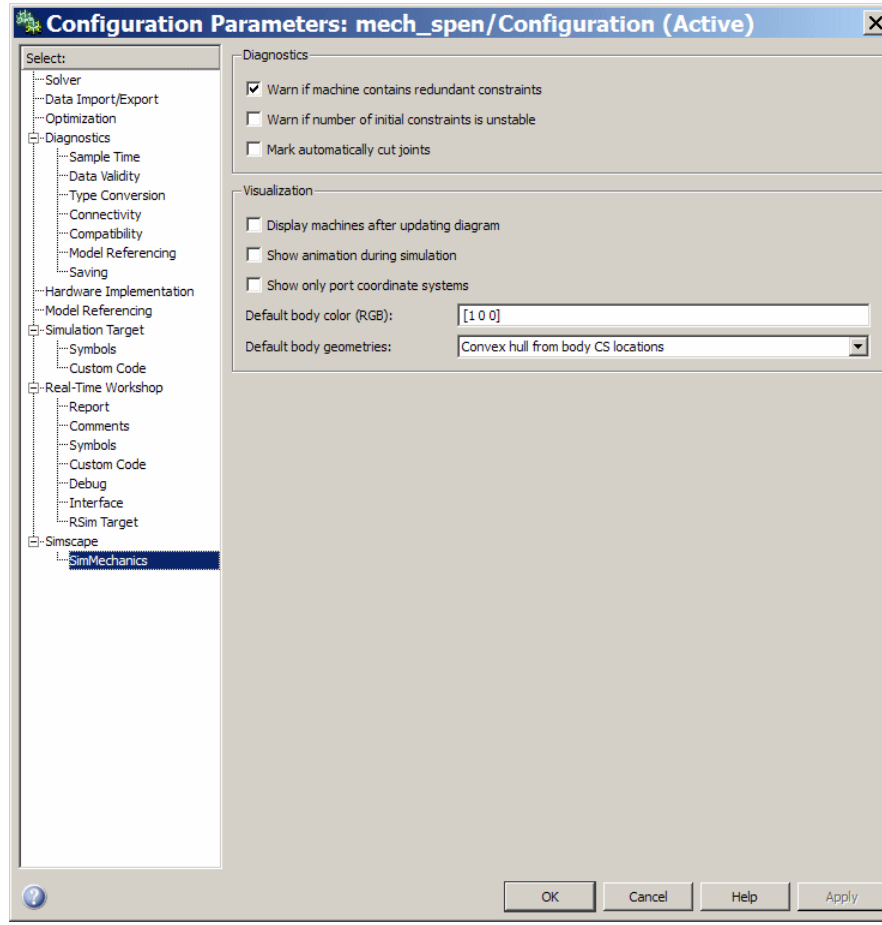
SimMechanics visualization displays a machine by displaying its bodies. You can display the bodies in two standard ways, by equivalent ellipsoids and by closed surfaces (convex hulls) enveloping the bodies' coordinate systems. This section explains how to visualize your pendulum using either standard body geometry.

You can view the pendulum before you start and, separately, choose to animate it during simulation as well.

Start Visualization

The first step is to configure the Configuration Parameters dialog.

1 On the Simulink menu bar, open the **Simulation** menu and select the **Configuration Parameters** entry. The Configuration Parameters dialog appears. Select the **SimMechanics** subnode, under the **Simscape** node, at the lower left.



2 To view the pendulum in its static initial state, select the **Display machines after updating diagram** check box.

To animate the pendulum visualization while the simulation is running, select the **Show animation during simulation** check box as well.

3 Click **OK**. Select **Update Diagram** from the **Edit** menu to open the visualization window.

Select a Body Geometry

The information that you use to specify body properties in a SimMechanics model is enough to display each body in a standard abstract shape. Without an external body geometry definition, SimMechanics visualization does not have enough information to display its full shape.

Equivalent Ellipsoids

A rigid body has a unique equivalent ellipsoid, a homogeneous solid ellipsoid with the same inertia tensor.

Because the rod has an axis of symmetry, the x-axis in this case, two of its three generalized radii are equal: $a_y = a_z$. The generalized radii of the equivalent ellipsoid are $a_x = \sqrt{5/3}(L/2) = 0.646$ m and $a_y = a_z = \sqrt{5}(r/2) = 1.12$ cm.

Convex Hulls

Each Body coordinate system (CS) has an origin point, and the collection of all these points, in general, defines a volume in space. The minimum outward-bending surface enclosing such a volume is the convex hull of the Body CSs.

You created the pendulum body with only two Body CSs, CG and CS1. The convex hull excludes the CG CS and thus, for the pendulum rod, is just the CS1 origin, a point.

Implementing Your Body Geometry Choice

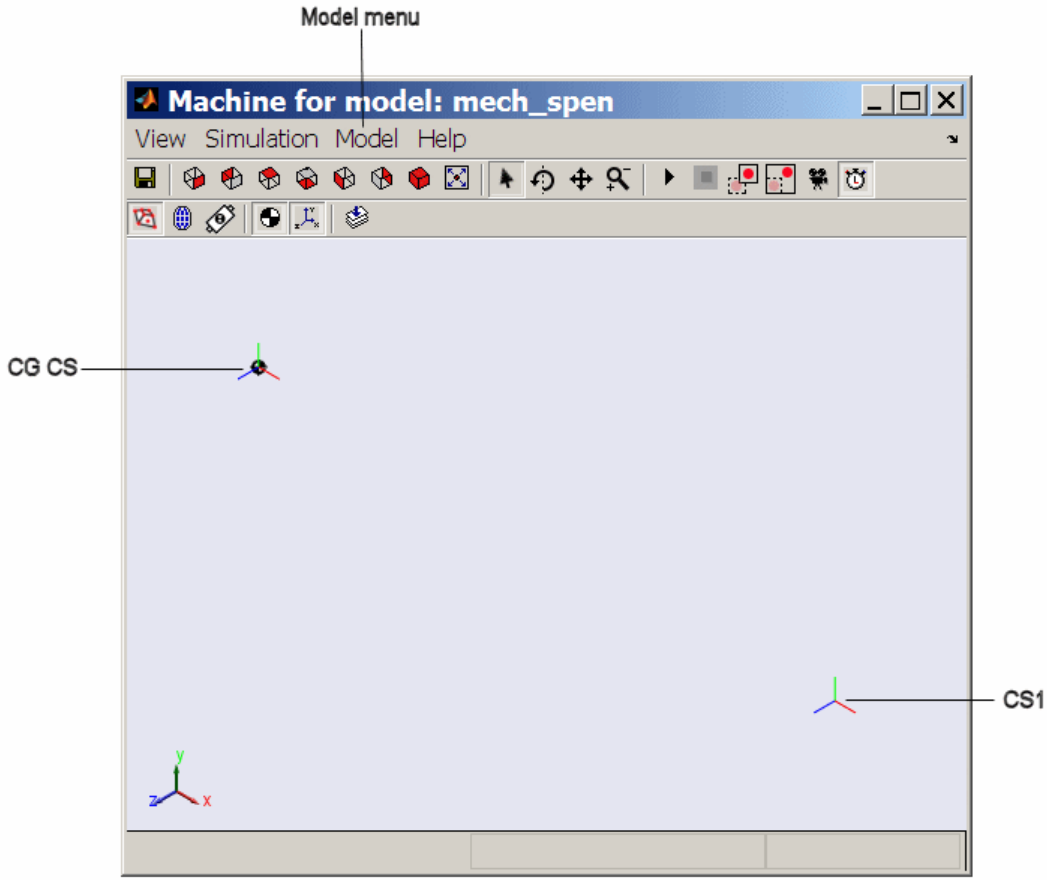
In the **Model** menu of the visualization window, you can choose how the pendulum or any machine bodies are displayed. In the **Body Geometries** submenu, choose **Convex Hulls** or **Ellipsoids**.

Display the Pendulum

You can access the SimMechanics visualization window from any SimMechanics model. To open it or to synchronize it at any time with your model, select **Update Diagram** in your model window's **Edit** menu.

Display the Pendulum as a Convex Hull

The displayed figure depends on the body geometry you choose. If you chose **Convex Hulls** in the **Model > Body Geometries** menu, a convex hull appears.



Pendulum Rod Displayed as a Convex Hull

You can change the viewpoint and manipulate the image with the controls in the toolbar and menus. Experiment with the **SimMechanics** menu's settings to see various ways of displaying the pendulum.

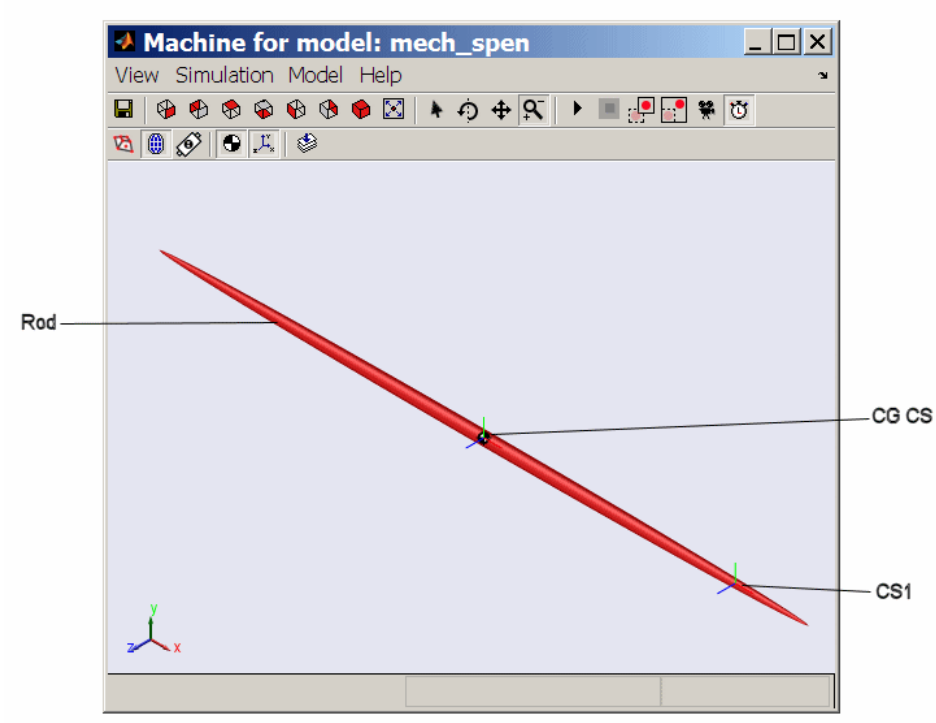
When you start the model, the body in the graphics window moves in step with the simulation.

Display the Pendulum as an Equivalent Ellipsoid

To display the pendulum as an equivalent ellipsoid, follow the previous steps, but change the body geometry choice:

1. Open the **Model** menu and select **Body Geometries**.
2. In the submenu, select **Ellipsoids**.

   The display changes. The equivalent ellipsoid looks like this.



Pendulum Rod Displayed as an Equivalent Ellipsoid

Model and Visualize More Complex Machines

The next tutorial shows how to create, run, and visualize a model for a more complex machine, a four bar mechanism. To configure Ground, Body, and Joint blocks now means repeating and expanding upon the three blocks of the first two tutorials.

Model and Simulate a Closed-Loop Machine

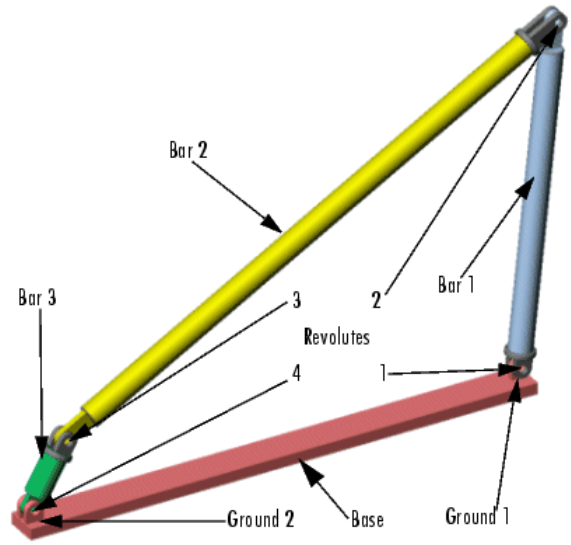
In this section...

Model the Four Bar Mechanism

In this tutorial, you build a model of a planar, four bar mechanism and practice using some of the important SimMechanics features.

You are urged to work through "Model and Simulate a Simple Machine" on page 2-11 and "Visualize a Simple Machine" on page 2-32 before proceeding with this section.

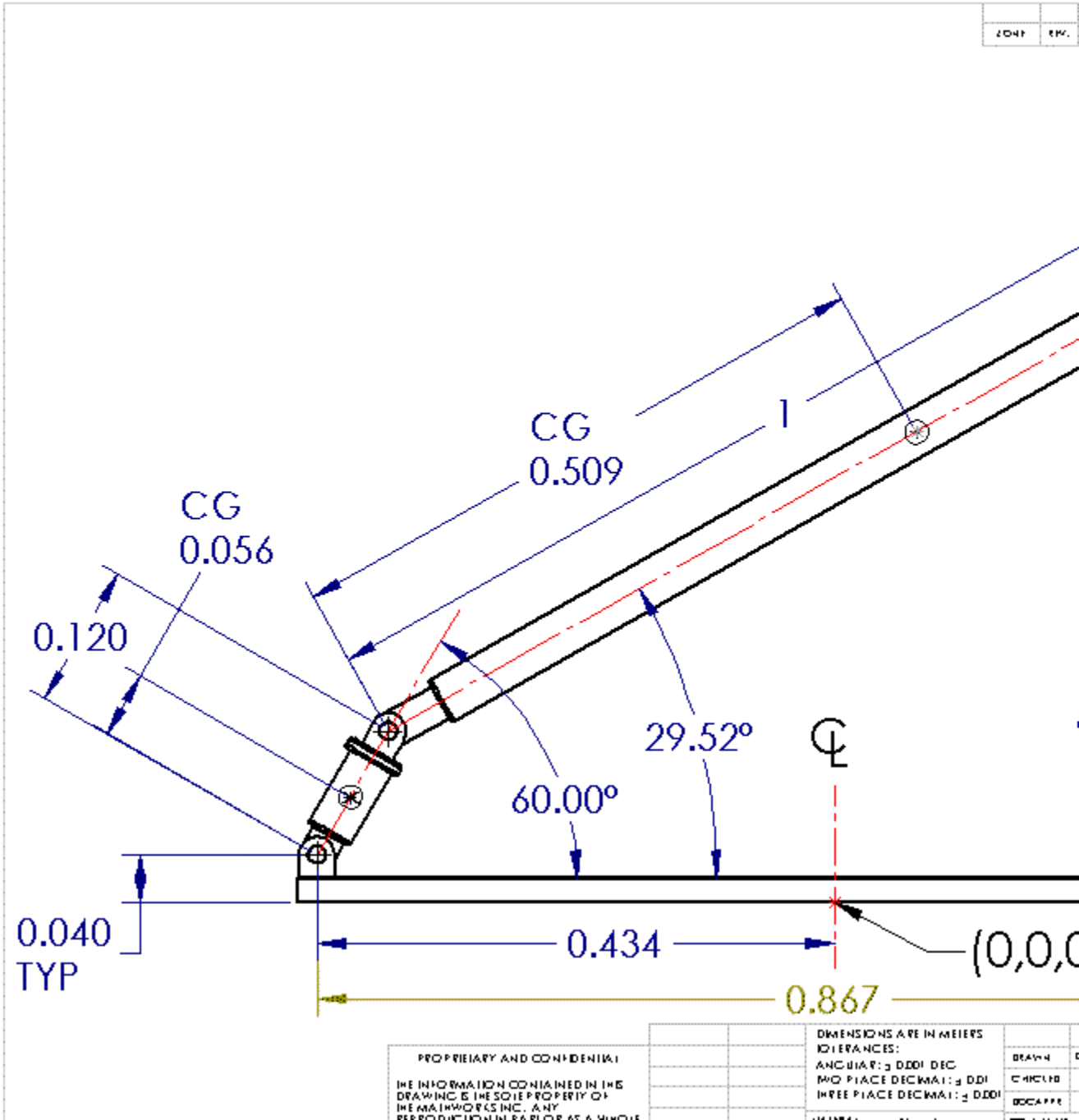
The system consists of three moving bars of homogeneous steel, two connected at one end each to ground points and a third crossbar connecting the first two. The base acts as an immobile fourth bar, with a Ground at each end. The mechanism forms a single closed loop, and its motion is confined to two dimensions.



A Four Bar Mechanism

The elementary parts of the mechanism are the bodies, while the revolute joints are the idealized rotational degrees of freedom (DoFs) at each body-to-body contact point. The bodies and the joints expressing the bodies' relative motions must be translated into corresponding SimMechanics blocks. If you want, you can add elaborations such as Constraints, Drivers, Sensors, and Actuators to this essential block diagram.

The following figure shows a detailed schematic of the four-bar mechanism.


CG
0.509
l
CG
0.056
0.120
29.52°
60.00°
0.040
TYP
0.434
0.867
(0,0,0
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF THE MATHWORKS INC. ANY
DIMENSIONS ARE IN METERS
TOLERANCES:
ANGULAR: ± 0.001 DEC
TWO PLACE DECIMAL: ± 0.01
THREE PLACE DECIMAL: ± 0.001
DRAWN
CHECKED
ENG APPR.

Count Degrees of Freedom

The three moving bars are constrained to move in a plane. So each bar has two translational and one rotational DoFs, and the total number of mechanical DoFs, before counting constraints, is 3*(2+1) = 9.

Because the motion of the bars is constrained, however, not all of these nine DoFs are *independent*:

- In two dimensions, each connection of a body with another body or with a ground point imposes two restrictions (one for each coordinate direction).

  Such a restriction effectively eliminates one of the two body ends as independently moving points, because its motion is determined by the next body's end.

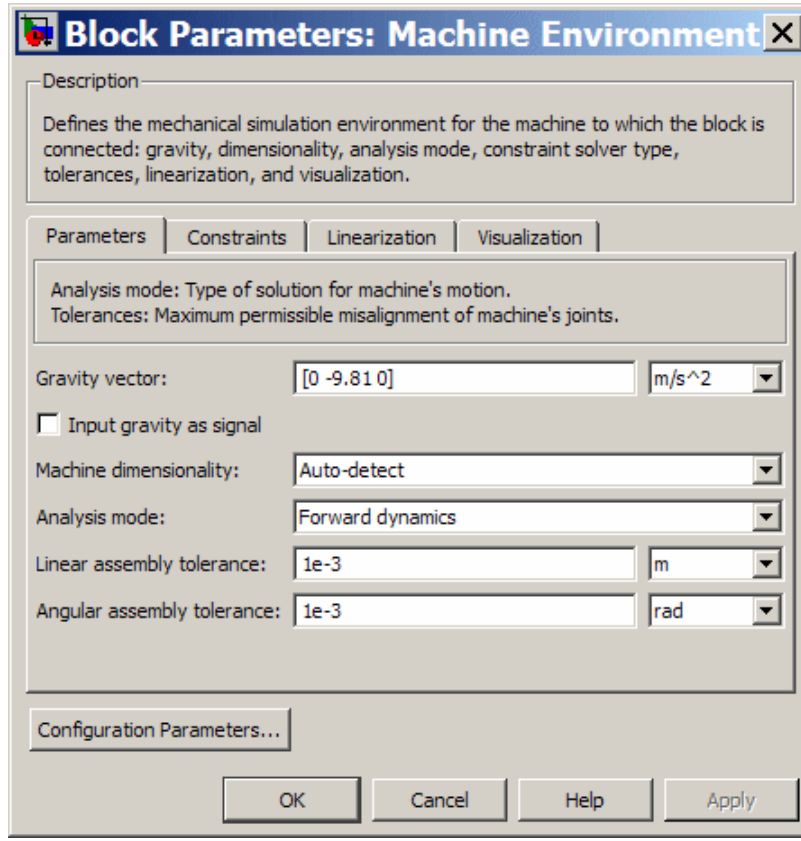
- There are four such body-body or body-ground connections and therefore eight restrictions implicit in the machine's geometry.

The eight restrictions on the nine apparent DoFs reduce the DoFs to one, 9 - 8 = 1. There are four rotational DoFs between bars or between bars and grounds. But three of these are dependent. Specifying the state of one rotational DoF fully specifies the other three.

Configure the Mechanical Environment

Open a new blank model window from the SimMechanics library. From the Bodies library, drag in and drop a Machine Environment block and a Ground block. Enable the Ground's Machine Environment port and connect the environment block to the Ground.

First you need to configure the machine's mechanical settings. Open the Machine Environment block. The block dialog box appears.



The Machine Environment Dialog Box Tabs

Click the four tabs in succession to display each pane.

Tab	Function
Parameters	Controls general settings for mechanical simulations
Constraints	Sets constraint tolerances and how constraints are interpreted
Linearization	Controls how SimMechanics models are linearized with Simulink
Visualization	Chooses whether or not to visualize the machine

Note some important features of this dialog box:

- The **Gravity vector** field specifies the magnitude and direction of gravitational acceleration and sets the vertical or up-down direction.
- The **Linear** and **Angular assembly tolerance** fields are also set here. Change **Angular assembly tolerance** to `1e-3 deg` (degrees). (See "Controlling Machine Assembly".)
- Leave the other defaults.

Close the dialog by clicking **OK**.

Starting Visualization

Tip If possible, open the visualization window before building a model. With it, you can keep track of your model components and how they are connected, as well as correct mistakes.

To visualize the bodies as you build the model, go to the **SimMechanics** node of the Configuration Parameters dialog:

1. Select the **Display machines after updating diagram** check box. If you want to animate the simulation later when you run the model, select the **Show animation during simulation** check box as well. Click **OK** or **Apply**.

   Then select **Update Diagram** from the **Edit** menu or enter **Crtl+D** at the keyboard. The visualization window opens.

2. In the **Model** menu, select **Body Geometries**, then **Ellipsoids**.

   As you add and change bodies in your model, you can update the display in your window at any time by updating your diagram.

Set Up the Block Diagram

In this set of steps, you create Bodies, position them, connect them with Joints, then configure the Body and Joint properties. The Body dialog boxes

give you many ways to represent the same system in the same physical state. This section explains one way.

Alternative, equivalent ways of configuring Bodies are discussed in "Body Coordinate Systems".

MAT-File Data Entry

The geometric and mass properties you need to specify for the Grounds and Bodies in this model are listed in the tables of the following two sections, "Configure the Ground and Joints" on page 2-46 and "Configure the Bodies" on page 2-50.

Instead of typing the numerical values of these properties into the dialog boxes, you can load the variable set you need into the workspace by entering

```
load fourbar_data
```

at the MATLAB command line. The variable name for each property is given in the tables. Just enter the appropriate variable names in the appropriate fields as you come to them in the dialog boxes.

Block Diagram Setup

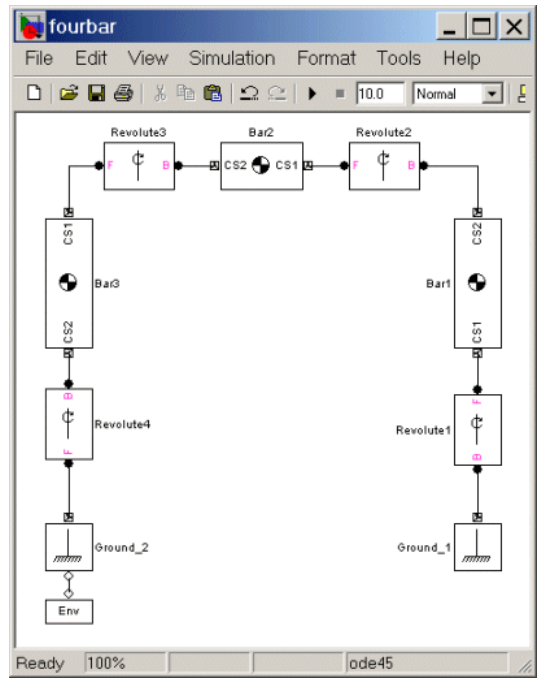
Your model already has one environment block and one ground block. Assemble the full model with these steps:

1. In the block library, open the Bodies library. Drag and drop another Ground block and three Body blocks into the new model window. Close the Bodies library.

2. From the Joints library, drag and drop four Revolute blocks into the model window.

3. Rotate and connect the blocks in the pattern shown in the following figure or with an equivalent block diagram topology.

   Use the block names shown in this figure for later consistency.



Connected Environment, Ground, Body, and Joint Blocks for the Four Bar

Block Diagram Topology. The topology of the block diagram is the connectivity of its elements. The elements are the Bodies and Grounds, connected by the Joints. Unlike the model of "Model and Simulate a Simple Machine" on page 2-11, the four bar mechanism is a closed-loop mechanism. The two Ground blocks represent points on the same absolute, immobile body, and they close the loop of blocks. The simple pendulum has only one ground and does not close its block connections.

To maintain consistent Body motion direction, make sure the Body coordinate system (CS) port pairs on each Body follow the sequence `CS1-CS2`, `CS1-CS2`, etc., for each bar, moving from Ground_1 to Ground_2, from right to left, as shown. To make the Joints consistent with the Body motion, the base-follower pairs `B-F`, `B-F`, etc., should follow the same right-to-left sequence.

Configure the Ground and Joints

Now configure the Ground blocks with the data from the following table. Grounded coordinate systems (CSs) are automatically created.

Geometry of the Four Bar Base

This table summarizes the geometry of ground points.

Geometric Properties of the Four Bar Grounds

Property	Value	MAT-File Variable
Ground_1 point (m)	`[ 0.433 0.04 0 ]`	`gpoint_1`
Ground_2 point (m)	`[-0.434 0.04 0 ]`	`gpoint_2`

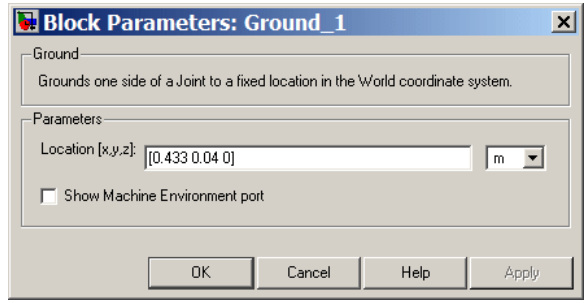
The base of the mechanism has these measurements:

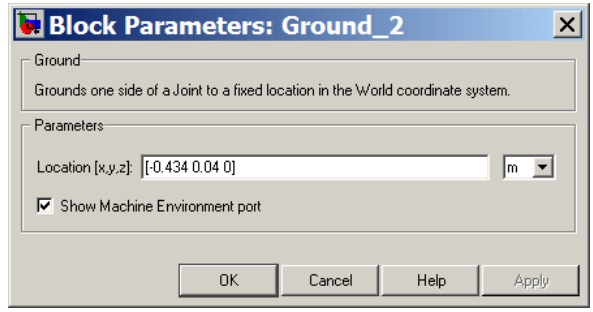
- The base is horizontal, with length 86.7 cm.
- Ground_1 represents the ground point 43.3 cm to the right of the World CS origin.
- Ground_2 represents the ground point 43.4 cm to the left of the World CS origin.
- The bottom revolutes are 4 cm above the origin (*x*-*z*) plane.

Set Up Grounds

To represent ground points on the immobile base, you need to configure the Ground blocks. Use the variable names if you've loaded `fourbar_data.mat` into your workspace:

1. Open Ground_1 and enter `[ 0.433 0.04 0 ]` or `gpoint_1` in the **Location** field.
2. Open Ground_2 and enter `[-0.434 0.04 0 ]` or `gpoint_2` in the **Location** field.
3. Leave both pull-down menus for units at default `m` (meters).





Configure the Revolute Joints

The three nongrounded bars move in the plane of your screen (*x*-*y* plane), so you need to make all the Revolute axes the *z*-axis (out of the screen):

1. Open each Revolute's dialog box in turn. In its **Parameters** area, note on the **Axes** tab that the *z*-axis is the default: **Axis of Action** is set to `[0 0 1]` in each, relative to **Reference CS** `World`. Leave these defaults.

   A Revolute block contains only one primitive joint, a single revolute DoF. So the **Primitive** is automatically `revolute`. Its name within the block is `R1`.

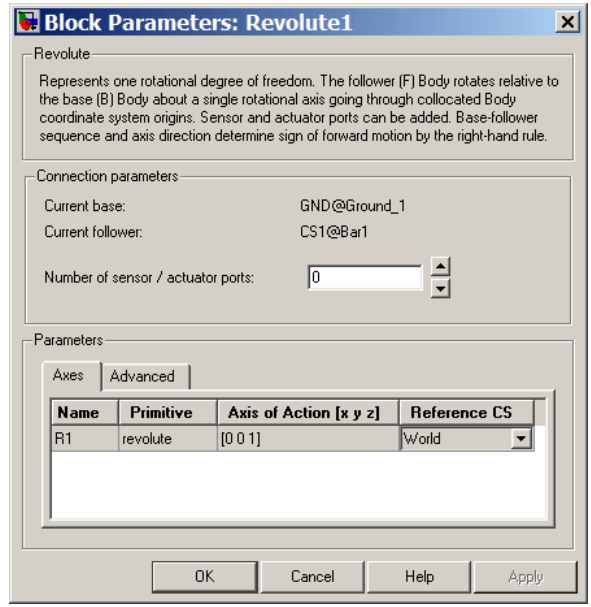
2. Leave these Revolute joint block defaults and ignore the **Advanced** tab.

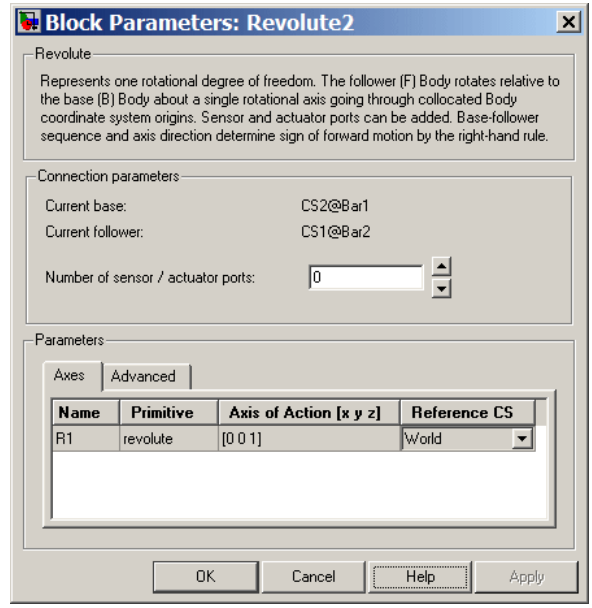
The Body CS and base-follower joint directionality should be set up as shown in the block diagram of the figure Connected Environment, Ground, Body, and Joint Blocks for the Four Bar on page 2-45. In the **Connection parameters** area, the default Joint directionality for each Revolute automatically follows the right-to-left sequence of Grounded and Body CSs:

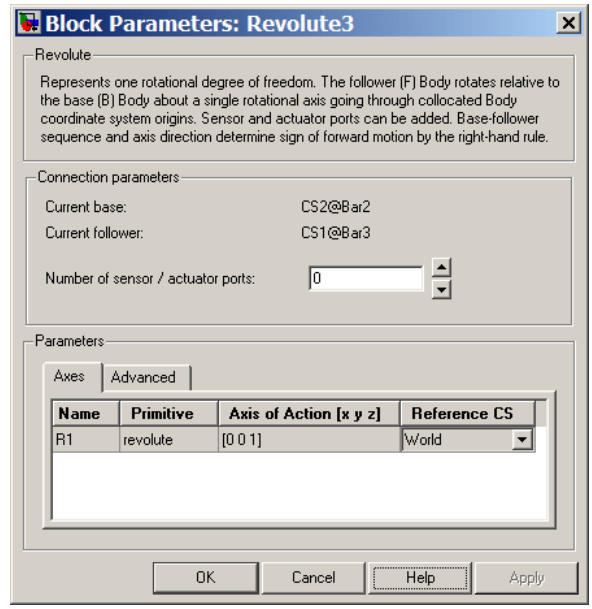
- Revolute1: Base to follower: `GND@Gound_1` to `CS1@Bar1`
- Revolute2: Base to follower: `CS2@Bar1` to `CS1@Bar2`
- Revolute3: Base to follower: `CS2@Bar2` to `CS1@Bar3`
- Revolute4: Base to follower: `CS2@Bar3` to `GND@Ground_2`

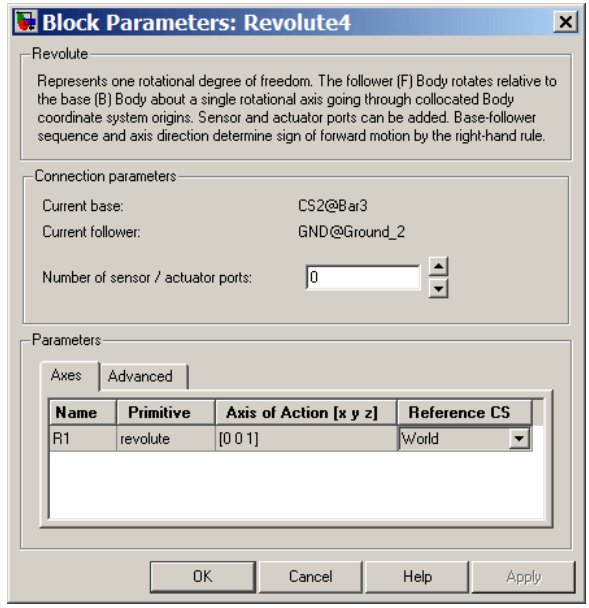
In this Joint directionality convention,

- At each Joint, the leftward Body moves relative to the rightward Body.
- The rotation axis points in the +*z* direction (out of the screen).
- Looking at the mechanism from the front in the figure, A Four Bar Mechanism on page 2-39, the positive rotational sense is counterclockwise. All Joint Sensor and Actuator data are interpreted in this sense.




Block Parameters: Revolute2
Revolute
Represents one rotational degree of freedom. The follower (F) Body rotates relative to the base (B) Body about a single rotational axis going through collocated Body coordinate system origins. Sensor and actuator ports can be added. Base-follower sequence and axis direction determine sign of forward motion by the right-hand rule.
Connection parameters
Current base:
CS2@Bar1
Current follower:
CS1@Bar2
Number of sensor / actuator ports:
0
Parameters
Axes
Advanced
Name
Primitive
Axis of Action [x y z]
Reference CS
R1
revolute
[0 0 1]
World
OK
Cancel
Help
Apply


Block Parameters: Revolute3
Revolute
Represents one rotational degree of freedom. The follower (F) Body rotates relative to the base (B) Body about a single rotational axis going through collocated Body coordinate system origins. Sensor and actuator ports can be added. Base-follower sequence and axis direction determine sign of forward motion by the right-hand rule.
Connection parameters
Current base:
CS2@Bar2
Current follower:
CS1@Bar3
Number of sensor / actuator ports:
0
Parameters
Axes
Advanced
Name
Primitive
Axis of Action [x y z]
Reference CS
R1
revolute
[0 0 1]
World
OK
Cancel
Help
Apply



Configure the Bodies

Setting the Body properties is similar for each bar, but with different parameter values entered into each dialog box:

- Mass properties
- Lengths and orientations
- Center of gravity (CG) positions
- Body coordinate systems (CSs)

In contrast to the first tutorial, where you specify Body CS properties with respect to the absolute World CS, in this tutorial, you specify Body CS origins on the bars in relative coordinates, displacing Bar1's CS1 relative to Ground_1, Bar2's CS1 relative to Bar1, and so on, around the loop. You can refer the definition of a Body CS to three types of coordinate systems:

- To World
- To the other Body CSs on the same Body

- To the Adjoining CS (the coordinate system on a neighboring body or ground directly connected by a Joint to the selected Body CS).

The components of the displacement vectors for each Body CS origin continue to be oriented with respect to the World axes. The rotation of each Body's CG CS axes is also with respect to the World axes, in the Euler X-Y-Z convention.

The following three tables summarize the body properties for the three bars.

Bar1 Mass and Body CS Data (MKS Units)

Property	Value	Variable Name
Mass	`5.357`	`m_1`
Inertia tensor	`[1.07e-3 0 0; 0 0.143 0; 0 0 0.143]`	`inertia_1`
CG Origin	`[0.03 0.282 0]` from `CS1` in axes of `World`	`cg_1`
CS1 Origin	`[0 0 0]` from `Adjoining` in axes of `World`	`cs1_1`
CS2 Origin	`[0.063 0.597 0]` from `CS1` in axes of `World`	`cs2_1`
CG Orientation	`[0 0 83.1]` from `World` in convention `Euler X-Y-Z`	`orientcg_1`

Bar2 Mass and Body CS Data (MKS Units)

Property	Value	Variable Name
Mass	`9.028`	`m_2`
Inertia tensor	`[1.8e-3 0 0; 0 0.678 0; 0 0 0.678]`	`inertia_2`
CG Origin	`[-0.427 -0.242 0]` from `CS1` in axes of `World`	`cg_2`

Bar2 Mass and Body CS Data (MKS Units) (Continued)

Property	Value	Variable Name
CS1 Origin	`[0 0 0]` from `Adjoining` in axes of `World`	`cs1_2`
CS2 Origin	`[-0.87 -0.493 0]` from `CS1` in axes of `World`	`cs2_2`
CG Orientation	`[0 0 29.5]` from `World` in convention `Euler X-Y-Z`	`orientcg_2`

Bar3 Mass and Body CS Data (MKS Units)

Property	Value	Variable Name
Mass	`0.991`	`m_3`
Inertia tensor	`[2.06e-4 0 0; 0 1.1e-3 0; 0 0 1.1e-3]`	`inertia_3`
CG Origin	`[-0.027 -0.048 0]` from `CS1` in axes of `World`	`cg_3`
CS1 Origin	`[0 0 0]` from `Adjoining` in axes of `World`	`cs1_3`
CS2 Origin	`[0 0 0]` from `Adjoining` in axes of `World`	`cs2_3`
CG Orientation	`[0 0 60]` from `World` in convention `Euler X-Y-Z`	`orientcg_3`

Configure the Bodies

Here are the common steps for configuring the Body dialogs of all three bars. See the three preceding tables for Body dialog box mass property (mass and inertia tensor) entries. The units are MKS: lengths in meters (m), masses in kilograms (kg), and inertia tensors in kilogram-meters^2 (kg-m^2).

1 Open all three Body dialogs for each bar. Enter the mass properties for each from the tables in the **Mass** and **Inertia** fields.

2 Now work in the Body coordinate systems area, the **Position** tab:

a Set the **Components in Axes of** menu, for each Body CS on each bar, to `World`.

b Leave units as default `m` (meters).

3 Set the Body CS properties for each Body CS on each bar from the data of the preceding tables:

a Enter the Body CS origin position data for CG, CS1, and CS2 on each bar from the tables or from the corresponding MAT-file variables.

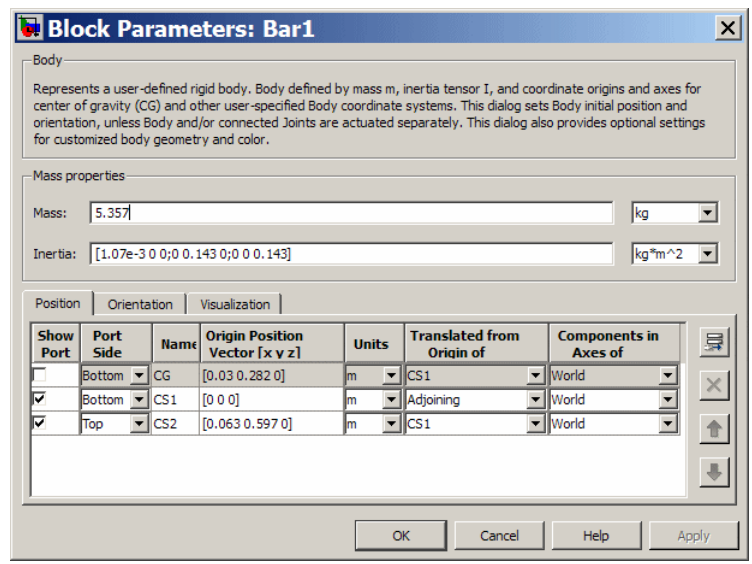
b Set the **Translated from Origin of** menu entries for each Body CS on each bar according to the values in the tables.

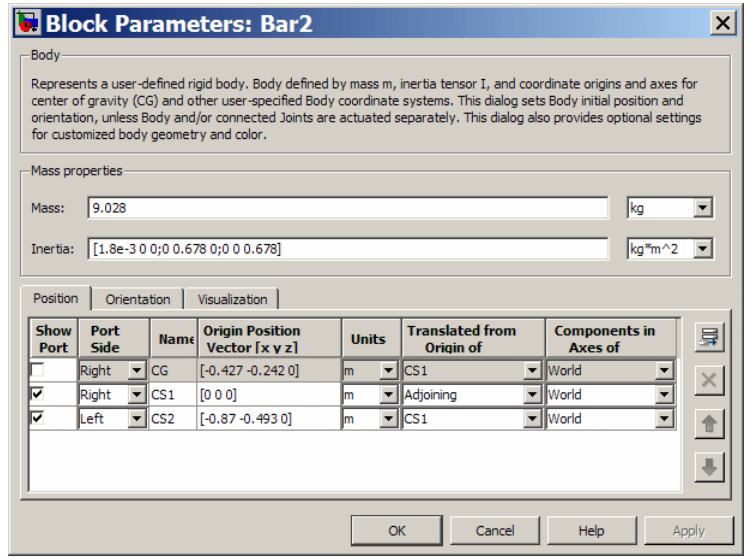
4 Select the **Orientation** tab by clicking its tab:

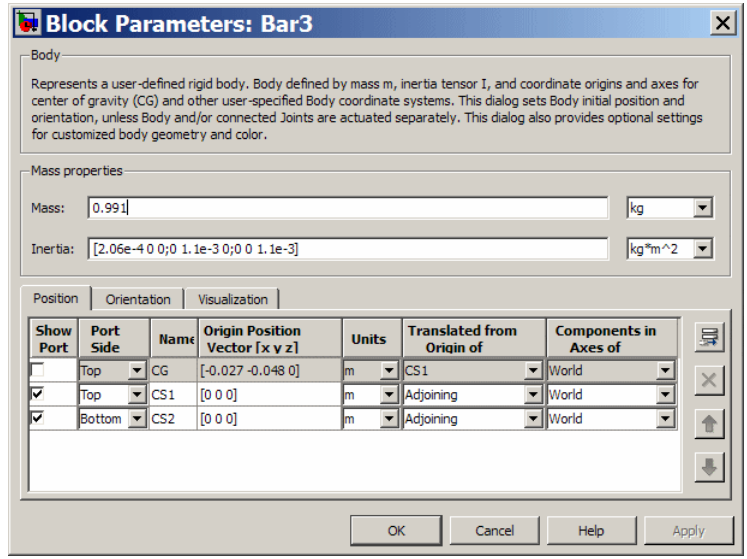
a Enter the **Orientation Vector** for the CG on each bar from the tables or from the corresponding MAT-file variables.

b Choose `World` for **Relative CS** in each case.

c Leave the other fields in their default values.

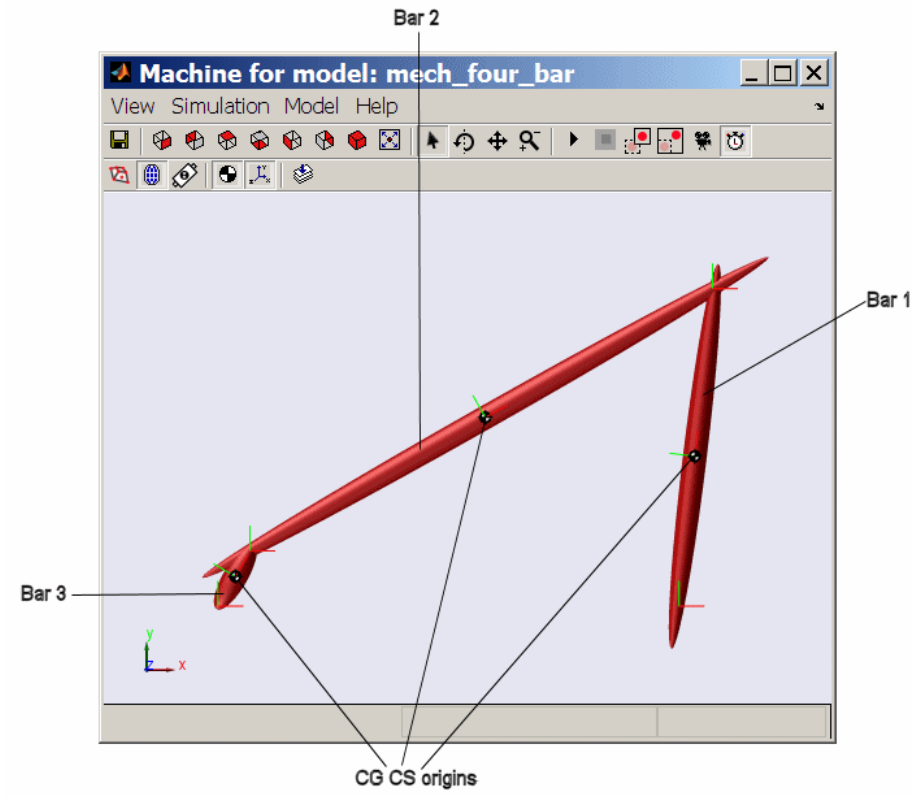



Block Parameters: Bar2
Body
Represents a user-defined rigid body. Body defined by mass m, inertia tensor I, and coordinate origins and axes for center of gravity (CG) and other user-specified Body coordinate systems. This dialog sets Body initial position and orientation, unless Body and/or connected Joints are actuated separately. This dialog also provides optional settings for customized body geometry and color.
Mass properties
Mass: 9.028
kg
Inertia: [1.8e-3 0 0;0 0.678 0;0 0 0.678]
kg*m^2
Position
Orientation
Visualization
Show Port
Port Side
Name
Origin Position Vector [x y z]
Units
Translated from Origin of
Components in Axes of
Right CG [-0.427 -0.242 0] m CS1 World
Right CS1 [0 0 0] m Adjoining World
Left CS2 [-0.87 -0.493 0] m CS1 World
OK
Cancel
Help
Apply


Block Parameters: Bar3
Body
Represents a user-defined rigid body. Body defined by mass m, inertia tensor I, and coordinate origins and axes for center of gravity (CG) and other user-specified Body coordinate systems. This dialog sets Body initial position and orientation, unless Body and/or connected Joints are actuated separately. This dialog also provides optional settings for customized body geometry and color.
Mass properties
Mass: 0.991
kg
Inertia: [2.06e-4 0 0;0 1.1e-3 0;0 0 1.1e-3]
kg*m^2
Position
Orientation
Visualization
Show Port
Port Side
Name
Origin Position Vector [x y z]
Units
Translated from Origin of
Components in Axes of
Top CG [-0.027 -0.048 0] m CS1 World
Top CS1 [0 0 0] m Adjoining World
Bottom CS2 [0 0 0] m Adjoining World
OK
Cancel
Help
Apply

Visualize the Bodies

The front view of the four bar mechanism, with the bodies displayed as equivalent ellipsoids, looks like this:



Sense Motion and Run Simulation

You finish building your model by setting initial conditions and inserting Sensors.

Before you start a simulation, you need to set its kinematic state or initial conditions. These include positions/angles and linear/angular velocities. This information, the machine's initial kinematic state, is discussed further in

"Kinematics and Machine Motion State" on page 3-2 and "Applying Motions and Forces".

You can sense motion in any model in two basic ways: sensing bodies or sensing joints. Here you sense Joint motion, using Joint Sensor blocks and feeding their Simulink signal outputs to Scope blocks.

Caution Because they are immobile, ground points cannot be moved, nor do they have any motion to measure.

Therefore, you cannot connect Ground blocks to Actuator or Sensor blocks.

Connect Joint Sensors

To sense the motion of the Revolute2 and Revolute3 blocks,

1. From the Sensors & Actuators library, drag and drop two Joint Sensor blocks into the model window. Drag Joint Sensor next to Revolute2 and Joint Sensor1 next to Revolute3.

2. Before you can attach a Joint Sensor block to a Revolute block, you need to create a new open round connector port ○ on the Revolute. Open Revolute2's dialog box:

   a. In the **Connection parameters** area in the middle, adjust the spinner menu **Number of sensor/actuator ports** to the value 1. Click **OK**.

      A new connector port ○ appears on Revolute2.

   b. Connect this connector port to the open round connector port on Joint Sensor.

3. Now repeat the same steps with Revolute3:

   a. Create one new connector port ○ on Revolute 3.

   b. Connect this port to Joint Sensor1.

4. Be sure to connect the outports > of the Sensor blocks to a Simulink Sink block. These outports are normal Simulink signals.

Plot Joint Motion

Here you can view the Joint Sensor measurements of Revolute2 and Revolute3's motions using a Scope block from the Simulink Sinks library:

1. Open the Simulink Library Browser. From the Sinks library, drag and drop a Scope block into your model window in between Joint Sensor and Joint Sensor1 blocks. Rename the Scope block "Angle."

2. Open the Angle block. In this scope window's toolbar, open the **Parameters** box. Under **Axes**, reset **Number of axes** to `2`. Click **OK**. A second inport > appears on the Angle block.

3. Expand the scope window for ease of viewing.

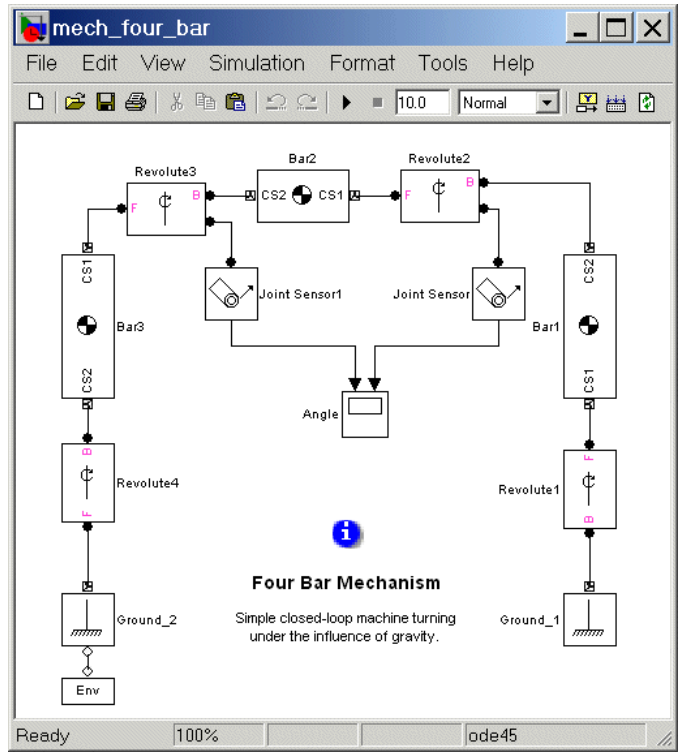
4. Connect the Joint Sensor and Joint Sensor1 block outports > to the Angle block inports >.

5. Open Joint Sensor and Joint Sensor1:

   a. In the **Measurements** area, **Connected to primitive** is set to `R1` in both blocks, indicating the first and only primitive revolute inside Revolute2 and Revolute3 to which each Sensor can be connected.

   b. Select the **Angle** check box to measure just the angle. Leave the units in default as `deg` (degrees). The Simulink line will contain one scalar.

Your completed model should look similar to the `mech_four_bar` example model.



Caution Sensor and Actuator blocks are the *only* blocks that can connect SimMechanics blocks to normal Simulink blocks.

Configure and Run Simulation

Now take the final steps to prepare and start the model:

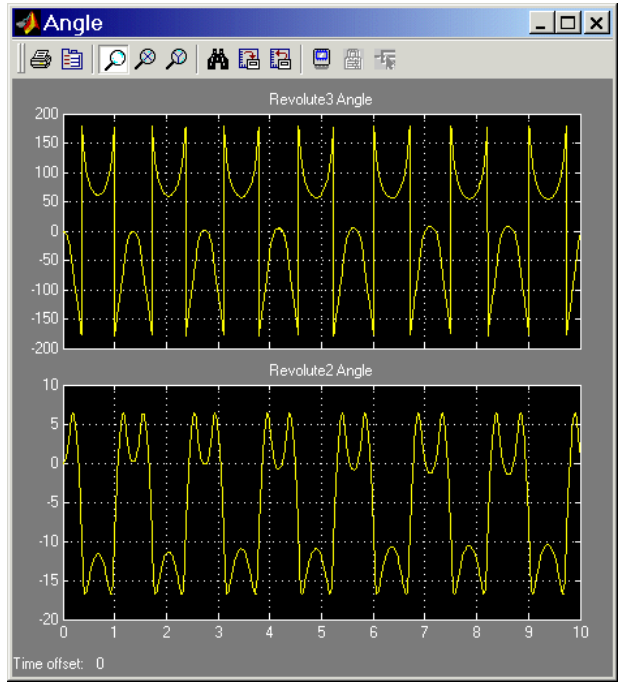
1. In the model window **Simulation** menu, select **Configuration Parameters**:
   a. In the **Solver** node, change **Absolute tolerance** to `1e-6`.

b Leave the other defaults and click **OK**.

2 Now run the model by clicking **Start** in the Simulink toolbar. The four bar mechanism will fall under the influence of gravity.

Note some features of the simulation:

- In this example, the mechanism starts from rest, with the initial velocities at zero. Thus the initial state of the four bar system is just the geometric state that you set up in "Set Up the Block Diagram" on page 2-43.
- The assembly at first falls over to the right, and the Revolute2 angle decreases.
- Bar3 turns all the way around, and Bar2 and Bar1 turn back to the left. The Revolute2 angle reverses direction. Revolute3 sweeps through a complete turn. Angles are mapped to the interval (-180^o,$+180^o$] and exhibit discontinuities.
- The motion repeats periodically, as there is no friction.



Animation

If you leave your visualization window open at the time you start the simulation and select the **Animate machine during simulation** check box in the **SimMechanics** node of the Configuration Parameters dialog, the visualized machine moves in step with the simulation.

You can now compare the animated motion with the Scope plots of the Revolute2 and Revolute3 angles.

3

Representing Motion

This chapter explains the SimMechanics representation of body position, orientation, and motion. It connects mechanics concepts commonly used in physics and engineering with specific SimMechanics implementations. The last section is a case study on configuring a SimMechanics Body block to represent position and orientation.

This chapter assumes some familiarity with mechanics and vector analysis. You should work through it as a single unit. Consult references for more information.

Kinematics and Machine Motion State

In this section...

About Kinematics

Kinematics is the description of a machine's motion without regard to forces, torques, and the mass properties of bodies. Because accelerations are proportional to forces and torques, if you know the mass properties of the bodies and the forces and torques applied to them, you need only the initial positions and their first derivatives (velocities) to integrate a machine's motion. You should also understand and keep in mind the following related concepts.

Degrees of Freedom

The relative position and orientation of a body with respect to a neighbor constitute up to six *degrees of freedom* (DoFs). The fundamental DoFs are translational (one body sliding relative to another along a prismatic axis) and rotational (one body rotating relative to another about a revolute axis, or one body pivoting relative to another about a spherical pivot point).

SimMechanics DoFs are represented by Joint blocks connected between Body blocks. Bodies without Joints have no DoFs and acquire DoFs only by having Joints connected to them. A SimMechanics machine's motion state is represented by the positions (prismatics), angles (revolutes or sphericals), and their first derivatives with respect to time (velocities).

The State of Motion

The *state of motion* of a mechanical system is the set of the instantaneous positions and orientations of all its bodies and their linear and angular velocities. SimMechanics body positions/orientations are *relative*: one body's

state is specified with respect to its neighbors. The absolute positions and velocities of the bodies' states are determined via the machine's connections to one or more grounds. These grounds are at rest in World, although they do not have to coincide with the World origin.

Home, Initial, and Assembled Configurations

When you start your model, the SimMechanics simulation configures your machines in preparation for motion by stepping them sequentially through three states.

- The simulation starts by analyzing the machines in their *home configurations*. A machine in its home configuration has all its bodies positioned and oriented purely according to the Body block dialog data. All body velocities are zero.
- From the model's initial condition actuators, the simulation then applies initial condition (position, orientation, and velocity) data to the joints of the model, changing its machines to their *initial configurations*.
- Finally, the simulation assembles any disassembled joints in the model, transforming the machines to their *assembled configurations*. While doing this, it holds fixed any positions and orientations specified by initial condition actuators.

  The assembled configuration is the final premotion machine state.

Updating your SimMechanics diagram (from the **Edit** menu or by pressing **Ctrl+D**) resets the model to its currently valid home configuration.

Representations of Body Motion

In this section...

About Body Motion

This section summarizes observer coordinate systems, measuring body motion, and the SimMechanics representation of a body's motion. It assumes a basic knowledge of vector algebra and analysis. In references, Goldstein; Murray, Li, and Sastry; and Shuster present coordinate transformations, rotations, and rigid body kinematics in detail. The preceding section, "Kinematics and Machine Motion State" on page 3-2, should also be helpful.

Each topic in this section builds on the preceding one. Therefore, you should scan linearly through the whole section, then read it in detail.

Machine Geometry and Motion

Machines are composed of bodies, which have relative degrees of freedom (DoFs). Bodies have positions, orientations, mass properties, and sets of Body coordinate systems. Joints represent the motions of the bodies.

- A machine's *geometry* consists of its static body features before starting a simulation: positions, orientations, and Body coordinate systems.
- A machine's *kinematics* consist of all degrees of freedom (DoFs) of all bodies: the positions/orientations and their derivatives of at any instant during the machine's motion.

The full description of a machine's motion includes not only its kinematics, but also specification of its observers, who define reference frames (RFs) and coordinate systems (CSs) for measuring the machine motion.

All vectors and tensors, unless otherwise noted, are represented by Cartesian matrices with three and nine, respectively, spatial components measured by rectangular coordinate axes.

Reference Frames and Coordinate Systems

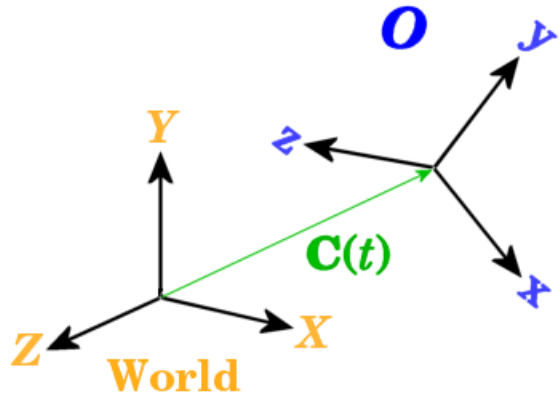
The *reference frame* of an observer is an observer's state of motion, which has to be measured by other observers. A SimMechanics model simulates a machine's motion using its Newtonian dynamics, which takes its simplest form in the set of *inertial* RFs, the set of all frames unaccelerated with respect to inertial space. Within an RF, you can pick any point as a *coordinate system* origin, then set up Cartesian (orthogonal) axes there.

The master SimMechanics inertial RF is called *World*. A CS origin and axis triad are also defined in World. World can mean either the RF or the CS, although in most contexts, it means the World coordinate system. World defines absolute rest and a universal coordinate origin and axes independent of any bodies and grounds in a machine.

A common synonym for coordinate system is *working frame*.

Relating Coordinate Systems in Relative Motion

Now add a second CS, called *O*, whose origin is translating with respect to the World origin and whose axes are rotating with respect to the World axes. Later in this section, this second CS is identified with a CS fixed in a moving body. (See "Representing Body Translations and Rotations" on page 3-9.)



A vector ***C*** represents the origin of *O*. Its head is at the *O* origin and its tail is at the World origin. The *O* origin moves as an arbitrary function of time ***C(t)***.

The orthogonal unit vectors {***u(x)***, ***u(y)***, ***u(z)***} define the coordinate axes of *O*.

- This set is oriented with respect to the World coordinate axes *X*, *Y*, *Z*, with unit vectors {***e(x)***, ***e(y)***, ***e(z)***}. The orientation changes with time.
- You can express the set {***u(x)***, ***u(y)***, ***u(z)***} as a linear combination of the basis {***e(x)***, ***e(y)***, ***e(z)***} in terms of nine coefficients. These are relationships between *vectors* (not vector *components*) and are independent of the reference frame and coordinate system.

$$\mathbf{u}(\mathrm{x}) = R_{\mathrm{xx}}\mathbf{e}(\mathrm{x}) + R_{\mathrm{yx}}\mathbf{e}(\mathrm{y}) + R_{\mathrm{zx}}\mathbf{e}(\mathrm{z})$$
$$\mathbf{u}(\mathrm{y}) = R_{\mathrm{xy}}\mathbf{e}(\mathrm{x}) + R_{\mathrm{yy}}\mathbf{e}(\mathrm{y}) + R_{\mathrm{zy}}\mathbf{e}(\mathrm{z})$$
$$\mathbf{u}(\mathrm{z}) = R_{\mathrm{xz}}\mathbf{e}(\mathrm{x}) + R_{\mathrm{yz}}\mathbf{e}(\mathrm{y}) + R_{\mathrm{zz}}\mathbf{e}(\mathrm{z})$$

- You obtain the *components* of the ***u***'s in World by projecting the ***u***'s on to the ***e***'s by scalar products. The time-dependent *R* coefficients represent the orientation of the ***u***'s with respect to the ***e***'s. You can use the labels (1,2,3) as equivalents for *(x,y,z)*.

$$u_x(x) = R_{xx}\ ,\ u_y(x) = R_{yx}\ ,\ u_z(x) = R_{zx}$$
$$u_x(y) = R_{xy}\ ,\ u_y(y) = R_{yy}\ ,\ u_z(y) = R_{zy}$$
$$u_x(z) = R_{xz}\ ,\ u_y(z) = R_{yz}\ ,\ u_z(z) = R_{zz}$$

- The *components* of any vector $\boldsymbol{v}$ measured in World are $\boldsymbol{e}(i)\cdot\boldsymbol{v}$. Represent them by a column vector, $\boldsymbol{v}_{World}$. The components of $\boldsymbol{v}$ in O are $\boldsymbol{u}(i)\cdot\boldsymbol{v}$. Represent them by a column vector, $\boldsymbol{v}_O$. The two sets of components are related by the matrix transformation $\boldsymbol{v}_{World} = R_{WO}\cdot\boldsymbol{v}_O$. The coefficients R form a matrix whose *columns* are the components of the $\boldsymbol{u}$'s in World:

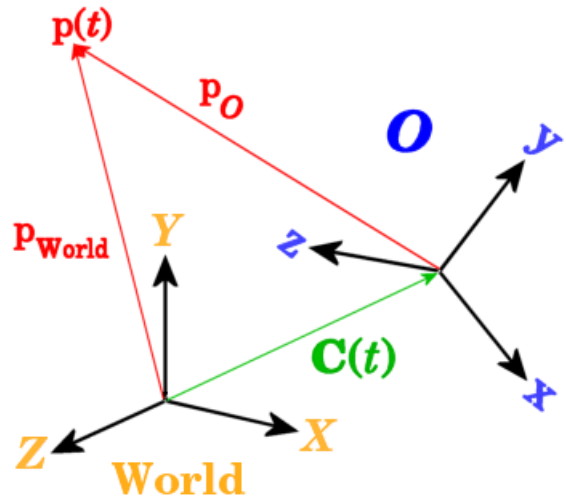
$$R = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} = \begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix}$$

  The orthogonality and unit length of the $\boldsymbol{u}$'s guarantee that R is an orthogonal rotation matrix satisfying $RR^T = R^TR = I$, the identity matrix. R^T is the transpose of R (switch rows and columns). Thus $R^{-1} = R^T$.

- Rotations always follow the *right-hand rule*, so that $\det(R) = +1$.
- You use rotation matrices in general to transform the components of any vector from one CS representation to another, rotated CS representation.

Observing Body Motion in Different Coordinate Systems

To the two observer CSs, World and O, now add a third point $\boldsymbol{p}$ in arbitrary motion. $\boldsymbol{p}$ could represent a point mass, the center of gravity (CG) of an extended body, or a point fixed in a moving rigid body, for example. The two observers describe the motion of this point in different ways, related to one another by time-dependent World-to-O coordinate transformations.



The components of $\boldsymbol{p}$ are given by projecting it on to some CS axes. The components of $\boldsymbol{p}$ as measured in World are a column vector $\boldsymbol{p}_{\text{World}}$ and, measured in O, are a column vector p_{O}. The two descriptions are related by

$$\boldsymbol{p}_{\text{World}} = \boldsymbol{C}_{\text{World}} + R \cdot \boldsymbol{p}_{\text{O}}$$

Thus the motion as measured by $\boldsymbol{p}_{\text{World}}$, when transformed and observed by O as p_{O}, has additional time dependence arising from the motion of C and R.

Relating Velocities Observed in Different Coordinate Systems

Differentiate the relationship between $\boldsymbol{p}_{\text{World}}$ and $\boldsymbol{p}_{\text{O}}$ once with respect to time. The result relates the velocity of $\boldsymbol{p}$ as measured by O to the velocity as measured in World.

$$\mathrm{d}\boldsymbol{p}_{\text{World}}/\mathrm{dt} = \mathrm{d}\boldsymbol{C}_{\text{World}}/\mathrm{dt} + R \cdot (\mathrm{d}\boldsymbol{p}_{\text{O}}/\mathrm{dt}) + (\mathrm{d}R/\mathrm{dt}) \cdot \boldsymbol{p}_{\text{O}}$$

The section “The Angular Velocity of a Body from Its Rotation Matrix” on page 3-9 explains how to express the third term in a simpler form.

Representing Body Translations and Rotations

Next consider the special case essential for describing the rigid body motions: the moving point $\boldsymbol{p}$ is fixed in the body itself. Let O be the center of gravity coordinate system (CG CS) of an extended rigid body (the origin of O at the CG itself) and let $\boldsymbol{p}$ be a point fixed somewhere in the same body. This body-fixed point is denoted by $\boldsymbol{b}$ in this special case. Because a moving body in general accelerates both translationally and rotationally, the CG CS is noninertial.

The rotation matrix R now describes the rotational motion of the body in terms of the rotation of the CG CS axes with respect to the World axes. Furthermore, because $\boldsymbol{b}$ is now fixed in the body itself, it does not move in O: $d\boldsymbol{b}_O/dt = 0$. All of its motion as seen by World is due implicitly to the motion of R and C.

The Angular Velocity of a Body from Its Rotation Matrix

Continue to identify O with the body CG CS and $\boldsymbol{b}$ as a point fixed in the body. The vector components of $\boldsymbol{b}$ are observed by World as $\boldsymbol{b}_{\text{World}}$ and by the CG CS as $\boldsymbol{b}_{\text{Body}}$. In the body, the point is immobile: $d\boldsymbol{b}_{\text{Body}}/dt = 0$. Its velocity observed by World is composed of the translational and rotational motion of the entire rigid body.

$$\mathrm{d}\boldsymbol{b}_{\text{World}}/\mathrm{dt} = \mathrm{d}\boldsymbol{C}_{\text{World}}/\mathrm{dt} + (\mathrm{d}R/\mathrm{dt})\cdot\boldsymbol{b}_{\text{Body}}$$

Because $RR^{\mathrm{T}} = I$, $(dR/dt)*R^{\mathrm{T}} + R(*dR^{\mathrm{T}}/dt) = 0$. Insert $R^{\mathrm{T}}R = I$ to the left of $\boldsymbol{b}_{\text{Body}}$ and define an antisymmetric matrix $\Omega = +(dR/dt)*R^{\mathrm{T}} = -R*(dR^{\mathrm{T}}/dt)$. Its components are $\Omega_{ik} = +\Sigma_j\, \varepsilon_{ijk}\omega_j$.

$$\Omega = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

where ω is the body's angular velocity in the World CS.

$$\begin{aligned}\mathrm{d}\boldsymbol{b}_{\text{World}}/\mathrm{dt} &= \mathrm{d}\boldsymbol{C}_{\text{World}}/\mathrm{dt} + \Omega\cdot R\cdot\boldsymbol{b}_{\text{Body}} \\ &= \mathrm{d}\boldsymbol{C}_{\text{World}}/\mathrm{dt} + \boldsymbol{\omega}\times(R\cdot\boldsymbol{b}_{\text{Body}})\end{aligned}$$

The motion of $\boldsymbol{b}_{\text{Body}}$ decomposes into the motion of the body's CG plus the angular rotation of $\boldsymbol{b}_{\text{Body}}$ relative to the CG, all measured in World.

The relationship between time derivatives of a vector measured in World and measured in the body holds generally. For any vector V,

$$(d\boldsymbol{V}/dt)_{\text{World}} = (d\boldsymbol{V}/dt)_{\text{Body}} + \boldsymbol{\omega}_{\text{World}} \times \boldsymbol{V}$$

The derivative of the angular velocity ω is the angular acceleration. It is the same, whether measured in World or in the body, because $\boldsymbol{\omega}\times\boldsymbol{\omega} = 0$.

The Permutation Symbol and the Vector Cross Product

The permutation symbol ε_{ijk} is defined by

ε_{ijk}= +1 if *ijk* is an even permutation (123 or any cyclic permutation thereof)

ε_{ijk}= -1 if *ijk* is an odd permutation (321 or any cyclic permutation thereof)

ε_{ijk} changes sign upon switching any two indices and vanishes if any two indices are equal. The components of the cross (vector) product $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b}$ of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are

$$c_i = \Sigma_{jk}\, \varepsilon_{ijk} a_j b_k$$

Representations of Body Orientation

In this section...

About Body Orientation Representations

You represent a SimMechanics body's orientation by specifying the orientation of its center of gravity coordinate system (CG CS) axes relative to some other set of axes, either the CS axes of an adjoining body or the World CS axes. No reorientation is represented by "no rotation" or the rotational identity.

A general rotation of a body in three dimensions has three independent degrees of freedom. There are many equivalent and interconvertible ways to represent these degrees of freedom . The Body and related Body Sensor and RotationMatrix2VR blocks use the following representations. The block reference pages for these blocks discuss block-specific details.

Axis-Angle Representation

The *axis-angle* representation of a rotation is the most fundamental form. Specify a rotation axis $\boldsymbol{n}$, then rotate by the right-hand rule about that axis by some angle θ. The vector $\boldsymbol{n} = (n_x, n_y, n_z)$ is a three-component unit vector, where $\boldsymbol{n} \cdot \boldsymbol{n} = n_x^2 + n_y^2 + n_z^2 = 1$. The axis $\boldsymbol{n}$ is sometimes called the *eigenaxis*.

SimMechanics models do not make direct use of the axis-angle representation, but it is the starting point for deriving other forms. It is also used extensively in mechanical applications such as computer-aided design and robotics.

The axis-angle representation is usually written as a 4-vector: $[n_x\ n_y\ n_z\ \theta]$. Of the four numbers, three are independent, because $\boldsymbol{n}$ always has unit length. The remaining freedom in this vector allows you to specify a direction (two angles) and the size and sense of the rotation about that directional axis (magnitude and sign of θ).

To describe continuous rotation in time, treat $\boldsymbol{n}$ and θ as functions of time.

Quaternion Representation

A *quaternion* represents a three-dimensional rotation as a four-component row vector of unit length:

$$q = \left[n_x \sin(\theta/2),\ n_y \sin(\theta/2),\ n_z \sin(\theta/2),\ \cos(\theta/2) \right] = [\boldsymbol{q}_v, q_s]$$

with $q*q = \boldsymbol{q}_v \cdot \boldsymbol{q}_v + q_s^2 = 1$. This definition uses the axis-angle representation defined above. The rotation angle about that axis is θ. To describe continuous rotation in time, treat $\boldsymbol{n}$ and θ as functions of time. Unlike some rotation representations, quaternions never become singular.

For more about quaternions, see Bell and Shuster in references.

Rotation Matrix Representation

The axis-angle representation also defines the *rotation matrix* R in exponential form $R = \exp(\theta\ \boldsymbol{n} \cdot \boldsymbol{J})$, where the J^k are real, antisymmetric matrices, and $\boldsymbol{n} \cdot \boldsymbol{J} = n_x J^1 + n_y J^2 + n_z J^3$. The rotation matrix R is orthogonal: $RR^T = R^T R = I$.

The J matrices are related to the antisymmetric permutation symbol ε_{ijk}.

$$\left(J^j\right)_{ik} = \varepsilon_{ijk}$$

The exponential R is reduced to closed form by the *Rodrigues identity*:

$$R = \exp(\theta \boldsymbol{n} \cdot \boldsymbol{J}) \ = \ I \ + \ (\boldsymbol{n} \cdot \boldsymbol{J}) \sin\theta \ + \ (\boldsymbol{n} \cdot \boldsymbol{J})^2 (1 - \cos\theta)$$

where I is the identity matrix, and $\boldsymbol{n} \cdot \boldsymbol{J}$ is given by

$$\mathbf{n} \cdot \boldsymbol{J} = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}$$

The inverse of R is identical to its transpose R^{T}. You can also obtain the inverse by replacing θ with θ or by reversing the direction of $\boldsymbol{n}$.

To describe continuous rotation in time, treat $\boldsymbol{n}$ and θ as functions of time.

Euler Angle Representation

An alternative representation for R is to rotate, in succession, about three independent axes, by three independent *Euler angles*. A full rotation R starting in *World* composes by multiplying the matrices successively on the *left*:

$$R_{\mathrm{BW}} = R_3 * R_2 * R_1$$

A full rotation R starting in a *body CS* composes by multiplying the matrices successively on the *right*:

$$R_{\mathrm{WB}} = R_1 * R_2 * R_3$$

The Euler angle convention is to

1. Rotate about one body coordinate axis (which rotates the other two).
2. Then rotate about a second body coordinate axis (rotated from its original direction) not identical to the first.
3. Lastly, rotate about another body coordinate axis not identical to the second.

Thus there are 3*2*2 = 12 possible Euler angle rotation sequences. The rotation axis sequences *Z-X-Z* and *Z-Y-X* are common. Rotation angles are often labeled as θ_1, θ_2, θ_3 or Φ, θ, Ψ as the first, second, and third angles, respectively. For example,

$$R_{\mathrm{BW}} = R_{\mathrm{X}}(\theta_1) * R_{\mathrm{Y}}(\theta_2) * R_{\mathrm{Z}}(\theta_3)$$

$$R_{WB} = R_Z(\Phi)*R_X(\theta)*R_Z(\Psi)$$

A two-dimensional rotation about a fixed axis requires one angle. For example, rotating the *x*- and *y*-axes about the *z*-axis by Φ is represented by

$$R_Z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

To describe continuous rotation in time, treat the Euler angles as functions of time. The Euler angle representation is singular in certain limiting situations. Such singularities are artifacts of the Euler angle form and have no geometric or physical significance.

Converting Rotation Representations

Certain SimMechanics blocks make use of different rotation representations.

- The Body block makes direct use of the Euler angle, rotation matrix, and quaternion representations.
- The Body Sensor block makes use of the rotation matrix.
- The RotationMatrix2VR block uses the rotation matrix and axis-angle forms.

The four rotation representations presented in this section are equivalent. You can represent a rotation equally well with any one of them. Some applications, however, tend to favor one representation over the others, and certain representations are singular in certain limits. It is helpful to know how to convert the various rotation representations into one another. The following summaries group the conversion formulas into one place.

Transforming the Axis-Angle Representation

The rotation axis unit vector $\boldsymbol{n}$ and the rotation angle θ define this representation, which is discussed in detail in “Axis-Angle Representation” on page 3-11. This representation defines the quaternion and rotation matrix representations:

$$q = \left[n_x \sin(\theta/2),\ n_y \sin(\theta/2),\ n_z \sin(\theta/2),\ \cos(\theta/2) \right] = [\boldsymbol{q}_\mathrm{v}, q_\mathrm{s}]$$

$$R = \exp(\theta \boldsymbol{n} \cdot \boldsymbol{J}) \ = \ I \ + \ (\boldsymbol{n} \cdot \boldsymbol{J}) \sin\theta \ + \ (\boldsymbol{n} \cdot \boldsymbol{J})^2 (1 - \cos\theta)$$

$$\mathbf{n} \cdot \boldsymbol{J} = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}$$

Transforming the Quaternion Representation

The quaternion is a vector-scalar pair, $q = [\boldsymbol{q}_\mathrm{v}\ q_\mathrm{s}]$, defined by "Quaternion Representation" on page 3-12. You can recover the axis-angle representation from the quaternion components:

$$\theta = 2 \cdot \cos^{-1}(q_\mathrm{s})$$

$$\mathbf{n} = \boldsymbol{q}_\mathrm{v} \Big/ \sqrt{1 - {q_\mathrm{s}}^2}$$

You can also construct the equivalent rotation matrix R from q.

$$R \ = \ (2q_\mathrm{s}^2 - 1)I + 2q_\mathrm{s}Q_\mathrm{v} + 2\boldsymbol{q}_\mathrm{v}^\mathrm{T} \otimes \boldsymbol{q}_\mathrm{v}$$

$$(Q_\mathrm{v})_{\mathrm{ik}} = \sum\nolimits_\mathrm{j} \varepsilon_{\mathrm{ijk}} (\boldsymbol{q}_\mathrm{v})_\mathrm{j}$$

The term $\boldsymbol{q}_\mathrm{v}^\mathrm{T} \otimes \boldsymbol{q}_\mathrm{v}$ is the *outer product* of $\boldsymbol{q}_\mathrm{v}$ with itself, the 3-by-3 matrix of $\boldsymbol{q}_\mathrm{v}$ components multiplied by each other.

Transforming the Rotation Matrix Representation

The rotation matrix R is an orthogonal 3-by-3 matrix: $RR^\mathrm{T} = R^\mathrm{T}R = I$, defined in "Rotation Matrix Representation" on page 3-12. You can invert the rotation matrix representation to obtain the equivalent representations for the quaternion $q = [\boldsymbol{q}_\mathrm{v}\ q_\mathrm{s}]$ and axis-angle ($\boldsymbol{n}$, θ)

$$q_s = \frac{1}{2}\sqrt{Tr(R)+1}$$
$$\boldsymbol{q}_v = Tr(\boldsymbol{J} * R) \Big/ \left(2\sqrt{Tr(R)+1}\right)$$
$$\theta = 2 \cdot \cos^{-1}\left(\frac{1}{2}\sqrt{Tr(R)+1}\right)$$
$$\mathbf{n} = Tr(\boldsymbol{J} * R) \Big/ \left(\sqrt{Tr(R)+1} \cdot \sqrt{3 - Tr(R)}\right)$$

The trace Tr of a matrix is the sum of its diagonal elements.

The J matrices constitute a 3-vector of matrices defined by the antisymmetric permutation symbol, $(J^j)_{ik} = \varepsilon_{ijk}$. See "The Permutation Symbol and the Vector Cross Product" on page 3-10 for more details.

The RotationMatrix2VR block converts the rotation matrix to the axis-angle representation.

Transforming the Euler Angle Representation

The Euler angle representation of a rotation, defined by "Euler Angle Representation" on page 3-13, stands apart from the other three, insofar as you cannot derive it from the axis-angle representation. It depends on the choice of rotation axis sequence, which generates multiple definition conventions. The Euler angle representation, at certain limits, can also be singular. Use caution with Euler angle expressions.

If you choose a convention and three angles, then compute R, you can convert R to the other representations by the use of "Transforming the Rotation Matrix Representation" on page 3-15 above. But given the nine components of R, you must find the Euler angles by inverting the nine equations that result from this matrix equation. (Only three equations of the nine are independent.) In some cases, angles can be read from R by inspection.

For example, choose rotations with respect to a Body coordinate system (CS) triad, in a commonly used rotation axis sequence Z-X-Z, with Φ, θ, Ψ as the respective angles. The rotation matrix is $R_{WB} = R_1(\Phi)*R_2(\theta)*R_3(\Psi)$,

$$R_{WB}(\phi,\theta,\psi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos\phi\cos\psi - \sin\phi\cos\theta\sin\psi & -\cos\phi\sin\psi - \sin\phi\cos\theta\cos\psi & \sin\phi\sin\theta \\ \sin\phi\cos\psi + \cos\phi\cos\theta\sin\psi & -\sin\phi\sin\psi + \cos\phi\cos\theta\cos\psi & -\cos\phi\sin\theta \\ \sin\theta\sin\psi & \sin\theta\cos\psi & \cos\theta \end{pmatrix}$$

In this convention, you can read θ from the R_{33} component, then find Ψ from the R_{32} or R_{31} component. Obtain Φ from one of the other components, using $\cos^2\Phi + \sin^2\Phi = 1$, or by multiplying from the right by $R_3\Psi^T$, then $R_2\theta^T$. The second method yields a unique solution for the sine and cosine of Φ.

Converting the Angular Velocity

The rotation matrix R is defined in "Representations of Body Motion" on page 3-4 and "Rotation Matrix Representation" on page 3-12.

The angular velocity vector ω is the rate at which a spinning CS rotates. R and the antisymmetric matrix Ω define ω:

$$\Omega = +(dR/dt)\cdot R^T = -R\cdot\left(dR^T/dt\right)$$

$$\Omega_{ik} = +\sum\nolimits_j \varepsilon_{ijk}\omega_j$$

$$\omega_j = \left(\tfrac{1}{2}\right)\sum\nolimits_{ik} \varepsilon_{ijk}\Omega_{ik}$$

You can also express the angular velocity in terms of Euler angles, by choosing a particular set of angles to represent R. See "Euler Angle Representation" on page 3-13 and "Transforming the Euler Angle Representation" on page 3-16.

The quaternion derivative is also related to the angular velocity:

$$d\boldsymbol{q}_v/dt = \left(\tfrac{1}{2}\right)\left(q_s\boldsymbol{\omega}_{Body} - \boldsymbol{q}_v \times \boldsymbol{\omega}_{Body}\right)$$

$$dq_s/dt = -\left(\tfrac{1}{2}\right)(\boldsymbol{q}_v \cdot \boldsymbol{\omega}_{Body})$$

Orienting a Body and Its Coordinate Systems

In this section...

About the Body Orientation Examples

This section shows you a set of examples of orienting a test body and its attached coordinate systems (CSs). It makes detailed use of the rotation representations and conversions explained in "Representations of Body Orientation" on page 3-11, and shows how to use MATLAB workspace variables in your SimMechanics block dialogs.

This sequence of examples assumes you are familiar with the basics of setting up and visualizing bodies and machines.

Setting Up the Test Body

The later examples require a configured body to work with. Here you set up a simple body with one Body coordinate system (CS), located at the center of gravity (CG).

Initializing the Body

First, create, connect, and visualize the initial body:

1. Open the SimMechanics block library and a new Simulink model window.
2. Drag and drop a Machine Environment, a Ground, and a Body block, as well as any Joint block you want, into your model window.
3. Open the Body dialog. In the Body coordinate systems area, delete the Body CS named CS2.

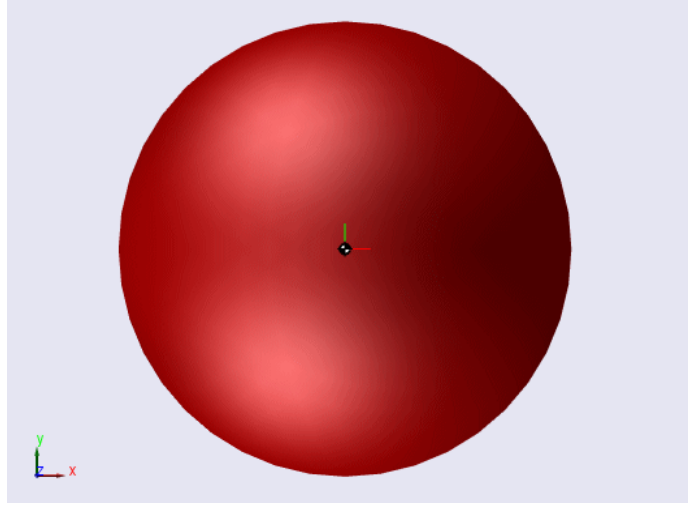
Leave the other defaults. The CG CS is located at **Origin position vector**`[0 0 0]`. Select **Show port** for the CG CS and deselect it for CS1. Delete the CS1 entry. Click **Apply**.

4 Open the Ground and select **Show Machine Environment port**. Click **OK**. Connect Ground to Machine Environment at the new Machine Environment port.

5 Connect the Body block at its CG CS port, through the Joint block, to Ground at its grounded CS port.

6 From the **Simulation** menu, open **Configuration Parameters** to the **SimMechanics** node. In the **Visualization** area, select the **Display machines after updating diagram** check box. Click **OK**.

7 From the **Edit** menu, select **Update Diagram**. When the window opens, open the **Model** menu and select **Body Geometries > Ellipsoids**. The window now displays the body as a sphere.



Configuring the Body

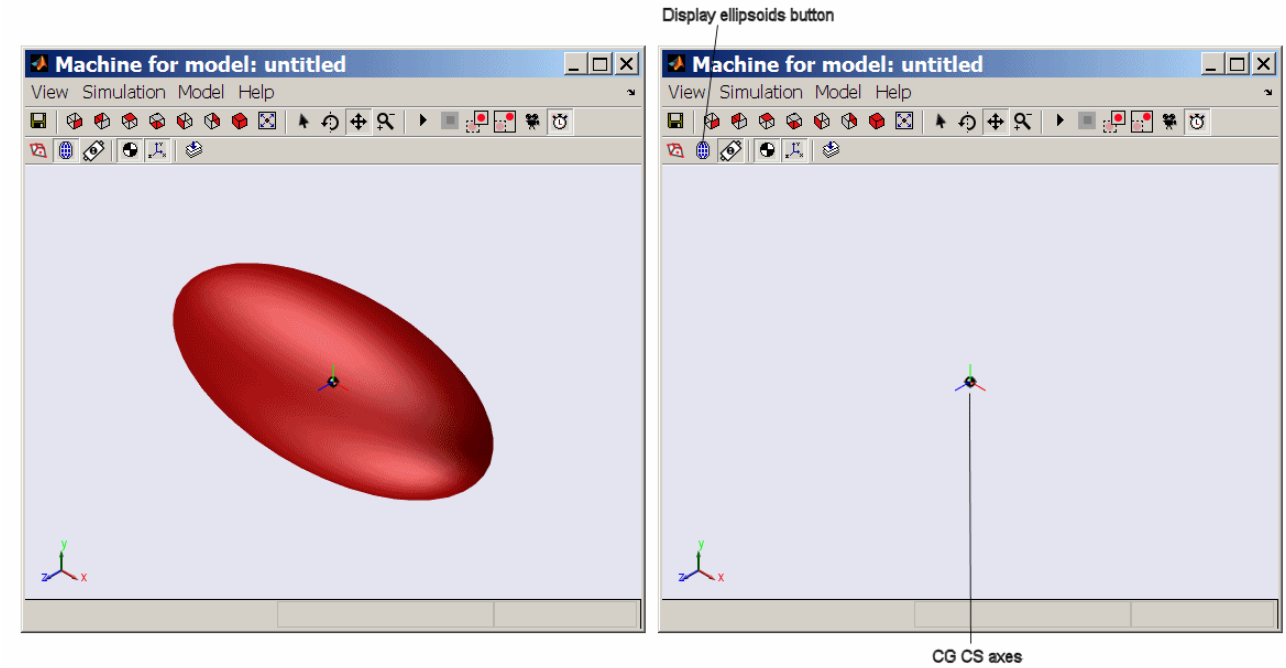
Now configure the body's mass and geometric properties:

1 In the Body dialog's **Mass properties** area, enter `11.5 kg` for the mass and `[18 0 0; 0 56 0; 0 0 56] kg*m^2` for the inertia. Click **Apply**.

2 Update your diagram. The body in the window changes to an ellipsoid, with *(x,y,z)* axes of length 2.02, 0.885, and 0.885 meters (m), respectively.

Switch from ellipsoid display to displaying individual body settings. This defaults back to convex hull display. But here, the CG CS triad alone displays. (There are no other CSs.) The *x*-, *y*-, and *z*-axes are red, green, and blue, respectively.

Turn the ellipsoid back on.



Test Body Ellipsoid: Initial Orientation, with Detail of CG CS Axes

Rotating the Body and Its CG CS Relative to World

In this example, you rotate the body, along with its CG CS axes, with respect to the World CS. The CG CS axes continue to have the same orientation with

respect to the body shape. The rotation is a positive turn of 75 deg (degrees) about the axis (1,1,1) in World. In this example, you rotate with a quaternion.

Computing the Rotation as a Quaternion

The unit vector ***n*** in the (1,1,1) direction is (1,1,1)/$\sqrt{3}$. The rotation angle is θ = 75 deg = 1.3090 rad. At the MATLAB command line, define `th = pi*75/180` and compute the quaternion components:

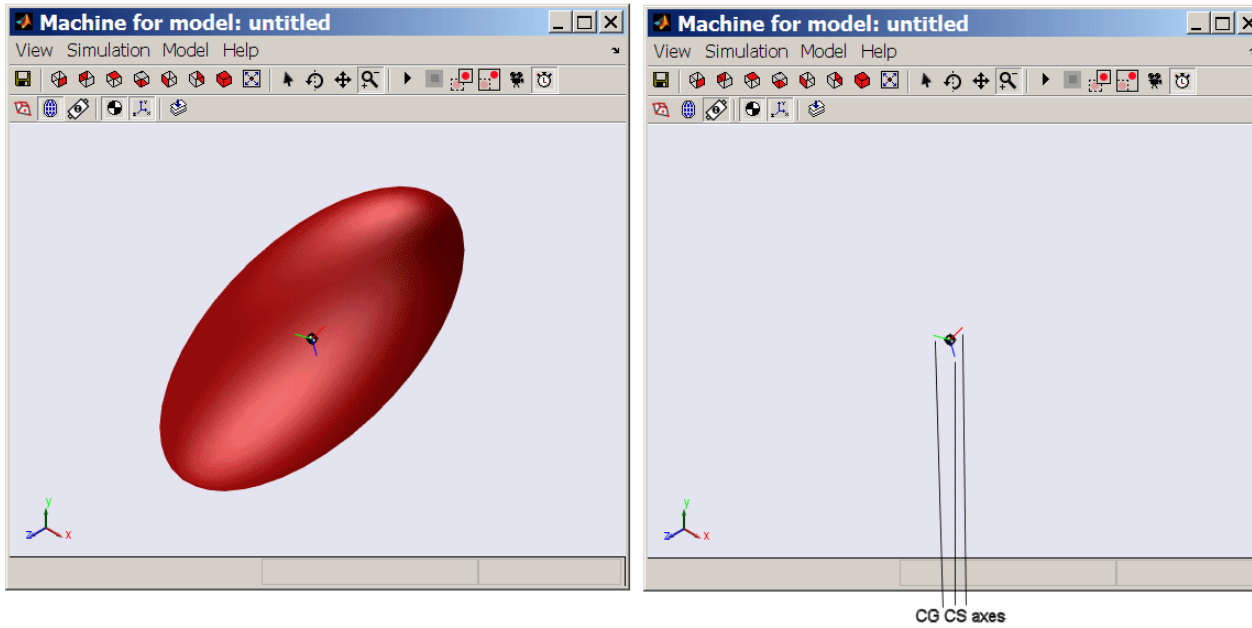
```
q = [sin(th/2)/sqrt(3) sin(th/2)/sqrt(3) sin(th/2)/sqrt(3) ...
cos(th/2)]

q =
    0.3515    0.3515    0.3515    0.7934
```

Rotating the Body and Its CG CS Axes with the Quaternion

To rotate the body and the CG CS coordinate axes together by this rotation,

1. In the Body dialog, click the **Orientation** tab in the Body coordinate systems area. Under the **Specified using convention** pull-down menu, select `Quaternion`.
2. Under **Orientation vector**, enter `q`. Leave the other defaults. The **Relative to coordinate system** field indicates that the rotation represented by `q` is oriented with respect to the World axes. Click **Apply**.
3. Update the diagram. The body and its CG CS rotate together relative to World:



Test Body Ellipsoid: Body and Its CG CS Axes Rotated Together

4 Finally rotate the body and its CG CS back to the original orientation by entering `[0 0 0 1]` under **Orientation vector** and clicking **Apply**. Update your diagram to refresh the visualization.

Rotating the Body Relative to Its Center of Gravity

You can also rotate the body *without* rotating its CG CS axes. To accomplish this requires leaving the Body CSs unchanged while rotating the body's inertia tensor relative to the CG CS. Here you use the same rotation as in the preceding example, but represent it as a rotation matrix.

Computing the Rotation as a Rotation Matrix

The unit vector $\boldsymbol{n}$ in the (1,1,1) direction is $(1,1,1)/\sqrt{3}$. The rotation angle is $\theta = 75$ deg = 1.3090 rad. Compute the rotation matrix:

```
nDotJ = [0 -1/sqrt(3) 1/sqrt(3); 1/sqrt(3) 0 -1/sqrt(3); ...
-1/sqrt(3) 1/sqrt(3) 0]
```

```
nDotJ =
         0   -0.5774    0.5774
    0.5774         0   -0.5774
   -0.5774    0.5774         0

R = eye(3) + nDotJ*sin(th) + nDotJ^2*(1-cos(th))
R =
    0.5059   -0.3106    0.8047
    0.8047    0.5059   -0.3106
   -0.3106    0.8047    0.5059
```

Rotating the Body's Inertia Tensor

The components of the inertia tensor that you enter into the Body dialog are always defined relative to the CG CS axes. If you hold the CG CS axes fixed and rotate the body by a rotation matrix R, the inertia tensor transforms according to $\boldsymbol{I}_{new} = R*\boldsymbol{I}_{old}*R^{T}$. Compute this with MATLAB:

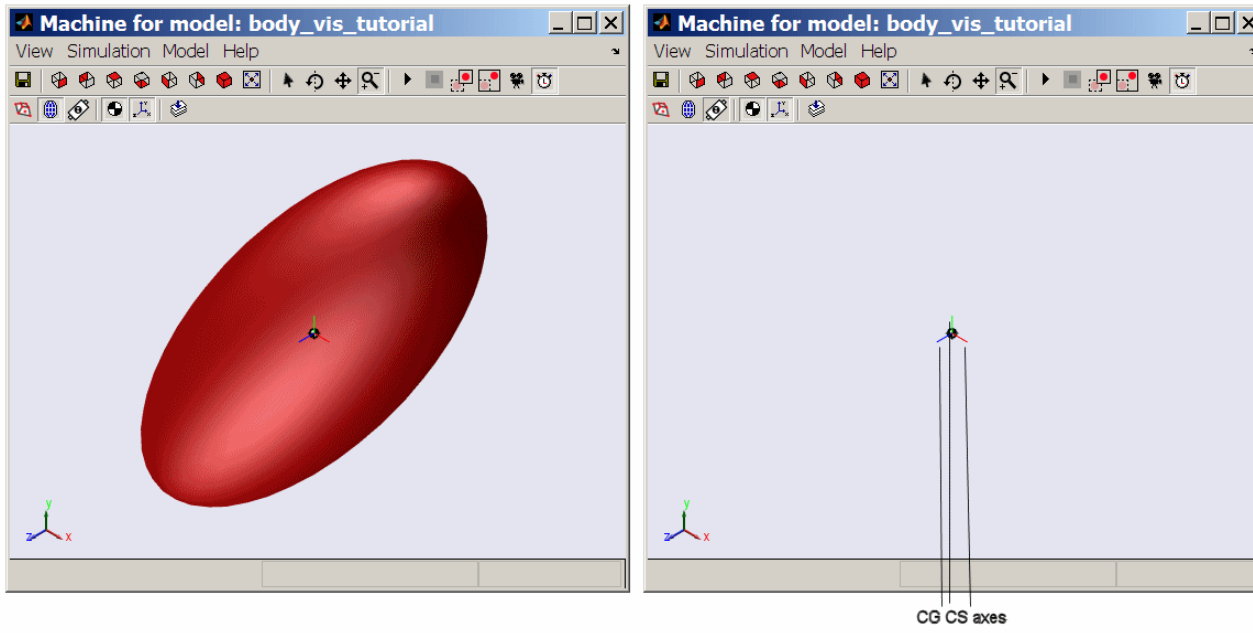
```
I = [18 0 0; 0 56 0; 0 0 56]
I =
    18     0     0
     0    56     0
     0     0    56

Irot = R*I*R'
Irot =
   46.2753  -15.4698    5.9711
  -15.4698   31.3911    9.4987
    5.9711    9.4987   52.3336
```

Rotating the Body with the Rotation Matrix

Symmetrize `Irot` by entering `Irot = (Irot + Irot')/2`. Then rotate the body alone relative to World and the CG CS.

1. In the Body dialog's **Mass properties** area, replace the existing inertia tensor with `Irot`. Click **Apply**.
2. Update the diagram. The body rotates again, as in the preceding example. But unlike that example, the CG CS axes do not change.



Test Body Ellipsoid: Body Rotated Relative to Its CG CS Axes

3 Finally rotate the body back to the original orientation by entering `I` in the **Inertia** field and clicking **Apply**. Update the diagram to refresh the visualization.

Creating and Rotating Body Coordinate Systems

In the preceding examples, you work with only one Body CS, the CG CS. In this example, you set up additional Body CSs and learn how to rotate them. It is common in mechanical applications to require extra Body CSs to locate sensors and actuators on bodies. Their axis orientations do not, in general, align with the orientation of the CG CS axes.

You visualize the body here using convex hulls, instead of ellipsoids, to articulate the Body CSs more clearly. Obtaining a full convex hull, with a surface enclosing a volume, requires at least four non-coplanar Body CSs.

This example also shows you how to obtain Euler angles and rotate with them.

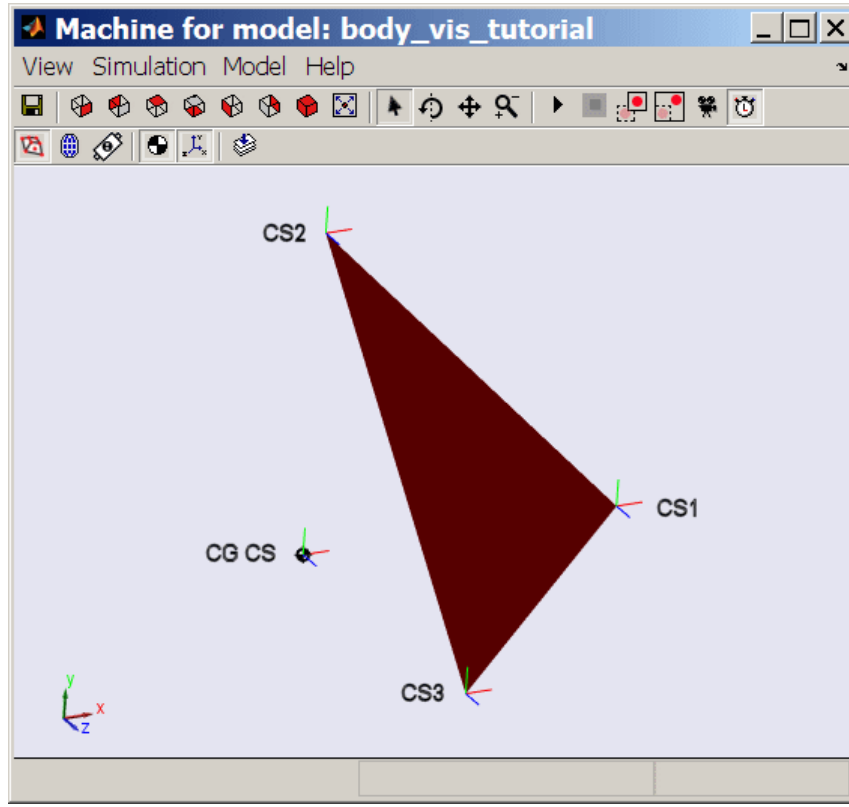
Creating and Viewing the New Body CSs

To change the visualization display to convex hulls,

1. Open the **Model** menu in the visualization window. Select the **Body Geometries** submenu.
2. Select **Convex Hulls**.

To create and visualize the new Body CSs,

1. Go to the Body dialog's Body coordinate systems area. Add three new coordinate systems to the CS list. Name them CS1, CS2, and CS3.
2. For these three CSs, change the **Origin position vector** fields to `[1 0 0]`, `[0 1 0]`, and `[0 0 1]`, respectively. Click **Apply**.
3. Update the diagram. The visualization window now display the body as a triangular surface. All the Body CS triads are oriented the same way, parallel to the CG CS axes and the World axes.



Test Body Convex Hull with CG CS and Three Body Coordinate Systems

Computing the Rotation as a Set of Euler Angles

The preceding examples used two rotation representations, the quaternion and the rotation matrix, based on the axis-angle representation. In this example, you use the same rotation as before, but represented as a set of Euler angles.

The rotation axis sequence convention is *Z-X-Z*, with Φ, θ, Ψ as the first, second, and third angles, respectively, the same as presented in “Converting Rotation Representations” on page 3-14. To obtain the angles, you equate the rotation matrix form for that convention:

$$R_Z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

to the numerical form of R you computed in "Rotating the Body Relative to Its Center of Gravity" on page 3-22:

```
R =
    0.5059   -0.3106    0.8047
    0.8047    0.5059   -0.3106
   -0.3106    0.8047    0.5059
```

and invert the resulting equations. The solution for θ and Ψ is

```
theta = acos(R(3,3))
theta =
    1.0404

psi = asin(R(3,1)/sin(theta))
psi =
   -0.3684
```

or about 60 and -21 deg, respectively.

Here is a method that yields a unique solution by using the structure of the Euler convention, $R = R_1(\Phi)*R_2(\theta)*R_3(\Psi)$. Multiply R on the right by $R_3(\Psi)^T*R_2(\theta)^T$, isolating $R_1(\Phi)$:

```
R3 = [cos(psi) -sin(psi) 0; sin(psi) cos(psi) 0; 0 0 1]
R3 =
    0.9329    0.3601         0
   -0.3601    0.9329         0
         0         0    1.0000

R2 = [1 0 0; 0 cos(theta) -sin(theta); 0 sin(theta) cos(theta)]
R2 =
    1.0000         0         0
         0    0.5059   -0.8626
         0    0.8626    0.5059
```

```
R1 = R*R3'*R2'
R1 =
    0.3601   -0.9329   -0.0000
    0.9329    0.3601   -0.0000
    0.0000    0.0000    1.0000

phi = acos(R1(1,1))
phi =
    1.2024
```

or about 69 deg.

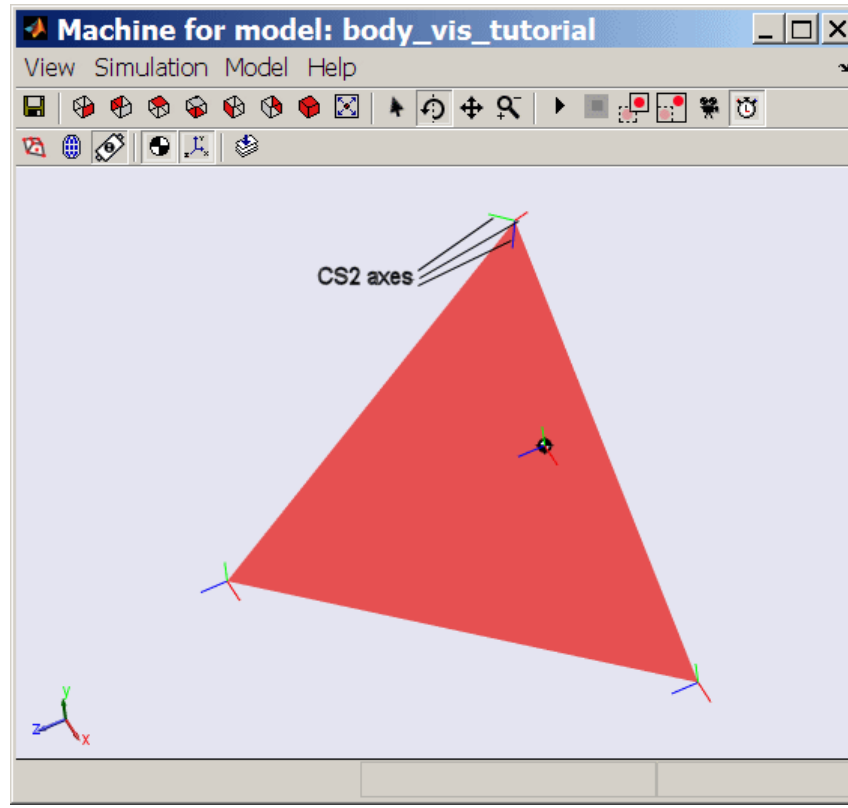
Rotating a Body CS Axis Triad

Here you rotate one of the Body CS axis triads, using the *Z-X-Z* rotation axis sequence convention and, for Euler angles, the Φ, θ, and Ψ you just found.

1 In the Body dialog's **Orientation** tab, locate the `CS2` entry.

   Under **Orientation vector**, enter `[phi theta psi]`. In the **Units** pull-down menu, select `rad`. In the **Specified using convention** pull-down menu, select `Euler Z-X-Z`. Click **Apply**.

2 Update your diagram. The CS2 axis triad, whose origin continues to be located at (0,1,0), now looks like this:



This rotated orientation of the CS2 axis triad is the same as that of the rotated CG CS in “Rotating the Body and Its CG CS Relative to World” on page 3-20. The two rotations are the same and produce the same result.

References

[1] Bell, E. T., "An Irish Tragedy: Hamilton (1805-1865)," in *Men of Mathematics,* New York, Simon & Schuster, 1937.

[2] Goldstein, H., *Classical Mechanics,* Second Edition, Reading, Massachusetts, Addison-Wesley, 1980.

[3] Murray, R. M., Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation,* Boca Raton, Florida, CRC Press, 1994.

[4] Shuster, M. D., "Spacecraft Attitude Determination and Control," in V. L. Piscane and R. C. Moore, eds., *Fundamentals of Space Systems,* New York, Oxford University Press, 1994.

A

Technical Conventions

Mechanical Conventions and Abbreviations

In this section...

Right-Hand Rule

For rotational motion and vector cross products $\boldsymbol{a} \times \boldsymbol{b}$, the right-hand (RH) rule is always assumed.

Vector Multiplication

Scalar-vector products and matrix-vector multiplication are denoted by $\boldsymbol{a} \cdot \boldsymbol{b}$ and $M\,\boldsymbol{v}$, respectively.

Common Abbreviations

These are the abbreviations of mechanical terms most commonly used in this guide.

Abbreviation	Meaning
CG	Center of gravity
CS	Coordinate system
DoF	Degree of freedom
RF	Reference frame

Glossary Terms

Special mechanical or SimMechanics terms are frequently hyperlinked online to entries in the "Glossary".

Mechanical Units

The SimMechanics environment accepts any mixture of meters-kilograms-seconds or MKS (SI), cgs, and English units.

Quantity	MKS (SI)	cgs	English
Length	meter (m)	centimeter (cm)	inch (in), foot (ft)
Time	second (s)	second (s)	second (s)
Mass	kilogram (kg)	gram (g)	slug (slug)
Velocity	meter/second (m/s)	centimeter/second (cm/s)	inch/second (in/sec), foot/second (ft/sec)
Acceleration (Gravity)	meter/second^2 (m/s^2)	centimeter/second^2 (cm/s^2)	inch/second^2 (in/sec^2), foot/second^2 (ft/sec^2)
Force	newton (N)	dyne (dyn)	pound (lb)
Angle	radian (rad), degree (deg)	radian (rad), degree (deg)	radian (rad), degree (deg)
Inertia	kilogram-meter^2 (kg-m^2)	gram-centimeter^2 (g-cm^2)	slug-foot^2 (slug-ft^2)
Angular velocity	radian/second (rad/s), degree/second (deg/s)	radian/second (rad/s), degree/second (deg/s)	radian/second (rad/sec), degree/second (deg/sec)
Angular acceleration	radian/second^2 (rad/s^2), degree/second^2 (deg/s^2)	radian/second^2 (rad/s^2), degree/second^2 (deg/s^2)	radian/second^2 (rad/sec^2), degree/second^2 (deg/sec^2)
Torque	newton-meter (N-m)	dyne-centimeter (dyn-cm)	pound-foot (lb-ft)

B

Bibliography

[1] Goldstein, H., *Classical Mechanics,* Second Edition, Reading, Massachusetts, Addison-Wesley, 1980.

[2] Goodman, L. E., and W. H. Warner, *Statics*, Mineola, New York, Dover Publications, 2001 (original edition, 1964).

[3] Goodman, L. E., and W. H. Warner, *Dynamics*, Mineola, New York, Dover Publications, 2001 (original edition, 1963).

[4] Haug, E. J., *Computer-Aided Kinematics and Dynamics of Mechanical Systems, Volume 1: Basic Methods,* Boston, Allyn & Bacon, 1989.

[5] José, J. V., and E. J. Saletan, *Classical Dynamics: A Contemporary Approach,* Cambridge, Cambridge University Press, 1998.

[6] JPL DARTS Web page on spatial operator algebra: http://dshell.jpl.nasa.gov/References/index.html.

[7] Marsden, J. E., and T. S. Ratiu, *Introduction to Mechanics and Symmetry,* Second Edition, New York, Springer-Verlag, 1999.

[8] Meriam, J. L., *Engineering Mechanics,* Fourth Edition, two volumes, New York, John Wiley and Sons, 1997.

[9] Murray, R. M., Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation,* Boca Raton, Florida, CRC Press, 1994.

[10] Von Schwerin, R., *MultiBody System SIMulation: Numerical Methods, Algorithms, and Software,* Berlin, Springer-Verlag, 1999.

Glossary

actuator

An *actuator* is a machine component that converts a Simulink signal into SimMechanics force, torque, or motion signals.

- You can configure a *body actuator* to apply forces/torques to a body either as an explicit function of time or through feedback forces/torques.
- You can configure a *joint actuator* to apply forces/torques between the bodies connected on either side of the joint.
- You can configure a *driver actuator* to apply relative motion between the bodies connected on either side of the driver.

Two specialized SimMechanics actuators set *joint initial conditions* and apply *stiction* to a joint, respectively.

Actuator	What the Actuator Does
Body Actuator	Applies forces to a body
Driver Actuator	Applies motion to a time-dependent constraint
Joint Actuator	Applies forces or motions to a joint
Joint Initial Condition Actuator	Sets a joint's initial conditions
Joint Stiction Actuator	Applies static and kinetic friction to joint motion

A SimMechanics Actuator block has an open round SimMechanics connector port ○ for connecting with a Body, Joint, or Driver block and an angle bracket > Simulink inport for connecting with normal Simulink blocks, such as Source blocks for generating force or torque signals.

See also *body*, *connector port*, *driver*, *initial condition actuator*, *joint*, *primitive joint*, *sensor,* and *stiction actuator*.

adjoining CS

The *adjoining CS* of a Body coordinate system (CS) is the CS on the neighboring body or ground directly connected to the original Body CS by a Joint, Constraint, or Driver.

See also *body*, *Body CS*, *coordinate system (CS)*, *grounded CS*, and *World*.

assembled configuration

A machine is in its *assembled configuration* once it passes from its home configuration through its initial configuration and its disassembled joints are then assembled. The assembly of joints can change body and joint configurations.

See also *home configuration* and *initial configuration*.

assembled joint

An *assembled joint* restricts the Body coordinate systems (CSs) on the two bodies at either end of the joint.

- For an *assembled prismatic joint,* the two Body CS origins must lie along the prismatic axis. The two Bodies translate relatively along the same axis.

  For an assembled joint with multiple prismatic primitives, the two Body CS origins must lie in the plane or space defined by the directions of the prismatic axes.

- For an *assembled revolute joint,* the two Body CS origins must be collocated. The two Bodies rotate relatively about the same axis.

  For an assembled joint with multiple revolute primitives, the two Body CS origins must be collocated.

- For an *assembled spherical joint,* the two Body CS origins must be collocated at the spherical primitive's pivot point. The two Bodies pivot relatively about this common origin.

You specify an assembly tolerance for assembled joints, the maximum dislocation distance allowed between all pairs of assembled Body CS origins and the maximum angle of misalignment between all pairs of assembled Body motion axes. If the distance dislocations and/or axis

misalignments in an assembled joint grow larger than the assembly tolerance, the simulation stops with an error.

See also *assembly tolerance, Body CS, collocation, disassembled joint, joint,* and *primitive joint*.

assembly

An *assembly* represents a mechanical system in computer-aided design (CAD). An assembly includes *parts* (bodies with full geometric, mass, and inertia tensor information), as well as *constraints* (sometimes called *mates*) restricting the *degrees of freedom* of the parts.

Every assembly has a *fundamental root* attached to the assembly coordinate origin. Assemblies can also have one or more *subassemblies* branching off the main assembly at a single point.

Assembly specifications typically also include design tolerances and how subassemblies move and how they are connected to the main assembly.

See also *associativity*, *body*, *computer-aided design (CAD)*, *constraint*, *degree of freedom (DoF)*, *fundamental root*, *inertia tensor*, *mass*, *part*, and *subassembly*.

assembly tolerance

The *assembly tolerance* is a range that determines how closely an *assembled joint* must be collocated and aligned. An *assembled joint* is connected on either side to Body coordinate systems (CSs) on two Bodies and restricts the relative configurations and motions of those Body CSs.

The assembly tolerances set the *maximum dislocation* of Body CS origins and *maximum misalignment* of motion axes allowed in assembled joints during the simulation.

- For *assembled prismatic primitives,* each pair of Body CS origins must lie in the subspace defined by the prismatic axes. Each pair of Bodies translates along these common axes.
- For *assembled revolute primitives,* each pair of Body CS origins must be collocated and their respective rotational axes aligned. Each pair of Bodies rotates about these common axes.
- For an *assembled spherical primitive,* the pair of Body CS origins must be collocated. The two Bodies pivot about this common origin.

A SimMechanics simulation attempts to assemble all joints in your machine at the start of simulation, including initially disassembled joints. If it cannot, the simulation stops with an error.

If the two Body CSs separate or the joint axes misalign in a way that makes their connecting assembled joint primitives no longer respect the assembly tolerances, the simulation stops with an error.

See also *assembled joint, Body CS, collocation, disassembled joint,* and *joint*.

associativity

Associativity is a persistent and session-independent parallel relationship among certain components of a CAD assembly, Physical Modeling XML files exported from it, and SimMechanics models generated from the XML. This relationship preserves the identities and parallelisms of certain CAD components (parts, geometric and kinematic relationships, subassembly hierarchy) and the corresponding components of the SimMechanics model. These SimMechanics model components include:

- Body and Ground blocks
- Joint blocks
- Coordinate systems connected to Joints
- Subsystem hierarchies

CAD assembly and SimMechanics model components related in this way are *associated*. If they are not so related, they are not associated or *nonassociated*.

See also *assembly*, *body*, *Body CS*, *computer-aided design (CAD)*, *constraint*, *ground*, *joint*, *part*, and *subassembly*.

axis-angle rotation

An *axis-angle rotation* mathematically represents a three-dimensional spherical rotation as a rotation axis vector $\boldsymbol{n} = (n_x, n_y, n_z)$ of unit length ($\boldsymbol{n} \cdot \boldsymbol{n} = n_x^2 + n_y^2 + n_z^2 = 1$) and a rotation angle θ. Define the rotation axis by the vector $\boldsymbol{n}$; rotate about that axis by θ using the right-hand rule. The $\boldsymbol{n}$ axis is sometimes called the *eigenaxis*.

The rotation axis direction is equivalent to specifying two independent angles; θ is the third independent angle making up the rotation.

In VRML, you represent body rotations by a vector signal [n_x n_y n_z θ].

See also *degree of freedom (DoF)*, *Euler angles*, *primitive joint*, *quaternion*, *right-hand rule*, *rotation matrix*, and *VRML*.

base (base body)

The *base body* is the body from which a joint is directed. The joint directionality runs from base to follower body.

Joint directionality sets the direction and the positive sign of all joint motion and force-torque data.

See also *body, directionality, follower (follower body),* and *right-hand rule*.

body

A *body* is the basic element of a mechanical system. It is characterized by its

- Mass properties (mass and inertia tensor)
- Position and orientation in space
- Attached Body coordinate systems

Bodies are connected to one another by joints, constraints, or drivers. Bodies carry no degrees of freedom.

You can attach to a Body block any number of Body coordinate systems (CSs). All SimMechanics Bodies automatically maintain a minimum of one Body CS at the body's center of gravity (CG). The Body block has distinctive axis triad CS ports, instead of the open, round connector ports, to indicate the attached Body CSs.

See also *actuator, adjoining CS, Body CS, center of gravity (CG), convex hull, coordinate system (CS), degree of freedom (DoF), equivalent ellipsoid, inertia tensor, joint, local CS, mass,* and *sensor*.

Body CS

A *Body CS* is a local coordinate system (CS) attached to a body, carried along with that body's motion. In general, bodies accelerate as they move, and therefore Body CSs define noninertial reference frames.

You can attach any number of Body CSs to a Body block, and you can choose where to place the Body CS origins and how to orient the Body CS axes. The Body block has distinctive axis triad Body CS ports instead of the open, round connector ports, to give you access to these Body CSs for connecting Joint, Sensor, and Actuator blocks.

Every Body block has an automatic, minimum Body CS at its center of gravity (CG). By default, it also has two other Body CSs for connection to adjacent Joints. Once you set the origin and axis orientation of each Body CS during Body configuration, the Body CSs are interpreted as fixed rigidly in that body during the simulation.

See also *body, center of gravity (CG), convex hull, coordinate system (CS), ground, grounded CS, local CS, reference frame (RF),* and *World*.

CAD

See *computer-aided design (CAD)*.

center of gravity (CG)

The *center of gravity* or center of mass of an extended body is the point in space about which the entire body balances in a uniform gravitational field. For translational dynamics, the body's entire mass can be considered as if concentrated at this point.

Every Body block has an automatic, minimum Body coordinate system (CS) with its origin at the CG — the CG CS. This origin point and the Body CS coordinate axes remain fixed rigidly in the body during the simulation.

See also *body, Body CS, degree of freedom (DoF), inertia tensor, kinematics,* and *primitive joint*.

CG

See *center of gravity (CG)*.

closed loop machine

A machine diagram contains one or more *closed loops* if, beginning at a starting point, you can trace a path through the machine back to the starting point without jumping out of or cutting the diagram. The number of closed loops is equal to the minimum number of cuttings needed to convert the diagram into a *tree* or *open* machine.

See also *open machine* and *topology*.

collocation

Collocation is the coincidence of two points in space, within assembly tolerances.

See also *assembled joint*, *assembly tolerance*, and *disassembled joint.*

composite joint

A *composite joint* is a joint compounded from more than one joint primitive and thus representing more than one degree of freedom. The joint primitives constituting a composite joint are the *primitives* of that joint.

A spherical primitive represents three rotational degrees of freedom, but is treated as a primitive.

See also *constrained joint, degree of freedom (DoF), joint,* and *primitive joint*.

computer-aided design (CAD)

Computer-aided design systems or platforms provide an environment to design machines, with full geometric information about *parts* (bodies) and their spatial relationships, as well as the *degrees of freedom* and mass properties of the parts.

A CAD representation of a machine is an *assembly*.

See also *assembly*, *associativity*, *body*, *constraint*, *degree of freedom (DoF)*, *inertia tensor*, *mass*, *part*, and *STL*.

connection line

You connect each SimMechanics block to another by using *SimMechanics connection lines*. These lines function only with SimMechanics blocks.

They do not carry signals, unlike normal Simulink lines, and cannot be branched. You cannot link connection lines directly to Simulink lines.

By default, connection lines appear red and dashed if they are not anchored at both ends to a connector port ○. Once you anchor them, the lines become black and solid.

However, if you have selected a nonnull choice in **Sample Time Display** from your model's **Format** menu, the connection line displays instead with whatever color chosen to indicate the sample time.

See also *actuator, connector port,* and *sensor*.

connector port

A *connector port* is an anchor for a connection line. Each SimMechanics block has one or more open round SimMechanics connector ports ○ for connecting to other SimMechanics blocks. You must connect these round ports only to other SimMechanics round ports. When an open connector port ○ is attached to a connection line, the Port changes to solid ●.

A *Connection Port block* is provided in the SimMechanics library to create a round SimMechanics connector port for an entire subsystem on that subsystem's boundary.

See also *actuator, connection line,* and *sensor*.

constrained joint

A *constrained joint* is a composite joint with one or more internal constraints restricting the joint's primitives.

An example is the Screw block, which has a prismatic and a revolute primitive with their motions in fixed ratio. Only one of these degrees of freedom is independent.

See also *degrees of freedom (DoF), joint,* and *primitive joint.*

constraint

A *constraint* is a restriction among degrees of freedom imposed independently of any applied forces/torques. A *constraint* removes one or more independent degrees of freedom, unless that constraint is *redundant* and restricts degrees of freedom that otherwise could not

move anyway. Constraints can also create *inconsistencies* with the applied forces/torques that lead to simulation errors.

- Constraints are *kinematic*: they must involve only coordinates and/or velocities. Higher derivatives of coordinates (accelerations, etc.) are determined by the Newtonian force and torque laws and cannot be independently constrained.
- Constraints are *holonomic* (integrable into a form involving only coordinates) or *nonholonomic* (not integrable; that is, irreducibly involving velocities).
- Constraints specify kinematic relationships that are explicit functions of time *(rheonomic)* or not *(scleronomic)*.

  SimMechanics *Constraints* represent scleronomic constraints, and *Drivers* represent rheonomic constraints. SimMechanics Constraint and Driver blocks are attached to pairs of Body blocks.
- In SimMechanics models with closed loops, one Joint, Constraint, or Driver per loop is internally cut and replaced by an invisible scleronomic or rheonomic constraint.
- Constraints are *redundant* if they independently impose the same restrictions on a machine.

In computer-aided design (CAD) assemblies, a constraint restricts one or more degrees of freedom of the assembly parts. (CAD constraints are sometimes called *mates*.) When a CAD assembly is converted to a SimMechanics model, such restricted degrees of freedom are translated into specific joints. (SimMechanics bodies have no degrees of freedom.)

See also *assembly*, *associativity*, *body*, *computer-aided design (CAD)*, *degree of freedom (DoF)*, *directionality*, *driver*, *joint*, *machine precision constraint*, *part*, *stabilizing constraint*, *tolerance*, and *topology*.

convex hull

The *convex hull* of a set of points in space is the surface of minimum area with convex (outward) curvature that passes through all the points in the set. In three dimensions, this set must contain at least four distinct, non-coplanar points to make a closed surface with nonzero enclosed volume.

The convex hull is an option for visualizing a SimMechanics body. The set of points is all the Body coordinate system (CS) origins configured in that Body block. The visualization of an entire machine is the set of the convex hulls of all its bodies. A SimMechanics convex hull excludes the body's center of gravity CS.

If a Body has fewer than four distinct, non-coplanar Body CSs, its convex hull is a lower-dimensional figure:

- Three distinct Body CSs produce a triangle without volume.
- Two distinct Body CSs produce a line without area.
- One Body CS produces a point without length.

See also *body*, *Body CS*, *equivalent ellipsoid*, and *STL*.

coordinate system (CS)

A *coordinate system* is a geometric object defined, in a particular reference frame, by a choice of *origin* and orientation of *coordinate axes*, assumed orthogonal and Cartesian (rectangular). An observer attached to that CS measures distances from that origin and directions relative to those axes.

SimMechanics software supports two CS types:

- World: *global* or *absolute inertial* CS at rest

  The assembly origin of a CAD assembly translated into a model becomes the World CS origin.

- Local:
  - Grounded CS
  - Body CS, including the center of gravity (CG) CS

    Constraint points on CAD parts in a CAD assembly translated into a model become Body CS origins on the bodies representing the parts.

Local coordinate systems are sometimes called *working frames*.

See also *body, Body CS, center of gravity (CG), computer-aided design (CAD)convex hull, grounded CS, local CS, reference frame (RF),* and *World*.

CS

A *coordinate system (CS).*

degree of freedom (DoF)

A *degree of freedom* is a single coordinate of relative motion between two bodies. Such a coordinate is free only if it can respond without constraint or imposed motion to externally applied forces or torques. For translational motion, a DoF is a linear coordinate along a single direction. For rotational motion, a DoF is an angular coordinate about a single, fixed axis.

A *prismatic* joint primitive represents a single translational DoF. A *revolute* joint primitive represents a single rotational DoF. A *spherical* joint primitive represents three rotational DoFs in angle-axis form. A *weld* joint primitive represents zero DoFs.

See also *body, coordinate system (CS), dynamics, joint,* and *kinematics*.

directionality

The *directionality* of a joint, constraint, or driver is its direction of forward motion.

The joint directionality is set by the order of the joint's connected bodies and the direction of the joint axis vector. One body is the *base* body, the other the *follower* body. The joint direction runs from base to follower, up to the sign of the joint axis vector. Reversing the base-follower order or the joint axis vector direction reverses the forward direction of the joint.

Joint directionality sets the direction and the positive sign of all joint motion and force-torque data.

Directionality of constraints and drivers is similar, except there is no joint axis, only the base-follower sequence.

See also *base (base body), body, follower (follower body), joint,* and *right-hand rule*.

disassembled joint

A *disassembled joint* is a joint that need not respect the assembly tolerances of your machine.

- For a *disassembled prismatic primitive*, the Body coordinate system (CS) origins do not have to lie on the prismatic axis.
- For a *disassembled revolute primitive*, the Body CS origins do not have to be collocated.
- For a *disassembled spherical primitive*, the Body CS origins do not have to be collocated.

A SimMechanics simulation attempts to assemble all disassembled joints in your machine at the start of simulation. If it cannot, the simulation stops with an error.

You can use disassembled joints only in a closed loop, with no more than one per loop.

See also *assembled joint*, *assembly tolerance, closed loop system*, *collocation*, and *topology*.

DoF

A *degree of freedom (DoF)*.

driver

A *driver* is a constraint that restricts degrees of freedom as an explicit function of time (a rheonomic constraint) and *independently* of any applied forces/torques. A driver removes one or more independent degrees of freedom, unless that driver is *inconsistent* with the applied forces/torques and forces a simulation error.

You specify the driver function of time in a dialog box in terms of an input Simulink signal from a Driver Actuator.

SimMechanics Driver blocks are attached to pairs of Body blocks.

See also *actuator*, *body*, *constraint*, *directionality*, and *degree of freedom (DoF)*.

dynamics

Dynamics is distinguished from *kinematics* by explicit specification of applied forces/torques and body mass properties.

A *forward dynamic* analysis of a mechanical system specifies

- The topology of how bodies are connected
- The degrees of freedom (DoFs) and constraints among DoFs
- All the forces/torques applied to the bodies
- The mass properties (masses and inertia tensors) of the bodies
- The initial condition of all DoFs:
  - Initial linear coordinates and velocities
  - Initial angular coordinates and velocities

The analysis then solves Newton's laws to find the system's motion for all later times.

Inverse dynamics is the same, except that the system's motion is specified and the forces/torques necessary to produce this motion are determined.

See also *constraint*, *degree of freedom (DoF)*, *inertia tensor*, *kinematics*, *mass*, and *topology*.

equivalent ellipsoid

The *equivalent ellipsoid* of a body is the homogeneous solid ellipsoid, centered at the body's center of gravity, with the same principal moments of inertia and principal axes as the body. A homogeneous solid ellipsoid is the simplest body with three distinct principal moments.

Every body has a unique equivalent ellipsoid, but a given homogeneous ellipsoid corresponds to an infinite number of other, more complicated, bodies. The rotational dynamics of a body depend only on its equivalent ellipsoid (which determines its principal moments and principal axes), not on its detailed shape.

The equivalent ellipsoid is an option for visualizing a SimMechanics body.

See also *body*, *convex hull*, *dynamics*, *inertia tensor*, *principal axes*, *principal inertial moments*, and *STL*.

Euler angles

Euler angles mathematically represents a three-dimensional spherical rotation as a product of three successive independent rotations about three independent axes by three independent (Euler) angles. Follow the Euler angle convention by

1. Rotating about one axis (which rotates the other two).
2. Then rotating about a second axis (rotated from its original direction) not identical to the first.
3. Lastly, rotating about another axis not identical to the second.

There are 3*2*2 = 12 possible Euler angle rotation sequences.

See also *axis-angle rotation*, *degree of freedom (DoF)*, *primitive joint*, *quaternion*, *right-hand rule*, and *rotation matrix*.

fixed part

A *fixed part* of a computer-aided design assembly or subassembly is a part that is welded to the assembly or subassembly root. A fixed part cannot move relative to the root.

See also *computer-aided design (CAD)*, *root body*, *subassembly*, and *subassembly root*.

follower (follower body)

The *follower body* is the body to which a joint is directed. The joint directionality runs from base to follower body.

Joint directionality sets the direction and the positive sign of all joint motion and force-torque data.

See also *base (base body), body, directionality,* and *right-hand rule*.

fundamental root

The *fundamental root* is a point in a computer-aided assembly that does not move, usually coincident with the assembly origin. All translational

and rotational motion of parts in the assembly reference this unmoving point.

The computer-aided assembly origin is recreated as the World coordinate system origin in a CAD-generated model.

See also *assembly*, *computer-aided design (CAD)*, *ground*, *part*, *root body*, and *World*.

ground

A *ground* or *ground point* is a point fixed at rest in the absolute or global inertial World reference frame.

Each ground has an associated grounded coordinate system (CS). The grounded CS's origin is identical to the ground point.

See also *body*, *coordinate system (CS)*, *grounded CS*, *machine*, and *World*.

grounded CS

A *grounded CS* is a local CS attached to a ground point. It is at rest in World, but its origin is wherever the ground point is and in general shifted with respect to the World CS origin. The coordinate axes of a grounded CS are always parallel to the World CS axes.

The World coordinate axes are defined so that:

+*x* points right

+*y* points up (gravity in -*y* direction)

+*z* points out of the screen, in three dimensions

You automatically create a Grounded CS whenever you set up a Ground block.

See also *adjoining CS*, *body*, *Body CS*, *coordinate system (CS)*, *ground*, *local CS*, and *World*.

home configuration

The bodies of a machine are in their *home configuration* when they are positioned and oriented purely according to the positions and

orientations entered into the Body dialogs. This configuration assumes zero body velocities.

See also *assembled configuration* and *initial configuration*.

inertia tensor

The *inertia* or *moment of inertia tensor* of an extended rigid body describes its internal mass distribution and the body's angular acceleration in response to an applied torque.

Let V be the body's volume and $\rho(\boldsymbol{r})$ its mass density, a function of vector position $\boldsymbol{r}$ within the body. Then the components of the inertia tensor $\boldsymbol{I}$ are:

$$I_{ij} = \int_V dV \left[\delta_{ij} |\boldsymbol{r}|^2 - r_i r_j \right] \rho(\boldsymbol{r})$$

The indices i, j range over 1, 2, 3, or x, y, z. This tensor is a real, symmetric 3-by-3 matrix or equivalent MATLAB expression.

The inertia tensor of a SimMechanics body is always evaluated in that body's center of gravity coordinate system (CG CS). That is, the origin is set to the body's CG and the coordinate axes are the CG CS axes.

Because the CG CS of a Body block is fixed rigidly in the body during simulation, the values of the inertia tensor components do not change as the body rotates.

See also *body, Body CS, equivalent ellipsoid*, *mass, principal axes,* and *principal inertial moments*.

initial condition actuator

An *initial condition actuator* sets a system's degrees of freedom nondynamically to prepare a system for dynamical integration, in a way consistent with all constraints.

The initial conditions are applied to a SimMechanics joint primitive.

See also *actuator, dynamics,* and *kinematics*.

initial configuration

A machine is in its *initial configuration* once all initial condition actuators have been applied to its joints. This step can change the

positions and orientations of the machine's bodies, as well as apply nonzero initial velocities.

See also *assembled configuration*, *home configuration*, and *initial condition actuator*.

joint

A *joint* is a machine component that represents one or more mechanical degrees of freedom between two bodies. Joint blocks connect two Body blocks in a SimMechanics schematic. A Joint has no mass properties such as a mass or an inertia tensor, and the reaction force and torque are equal and opposite on its two connected Bodies.

A *joint primitive* represents one translational or rotational degree of freedom or one spherical (three rotational degrees of freedom in angle-axis form). Prismatic and revolute primitives have motion axis vectors. A weld primitive has no degrees of freedom.

A *primitive joint* contains one joint primitive. A *composite joint* contains more than one joint primitive.

Joints have a directionality set by their base-to-follower Body order and the direction of the joint primitive axis. The sign of all motion and force-torque data is determined by this directionality.

See also *actuator*, *assembled joint*, *base (base body)*, *body*, *composite joint*, *constrained joint*, *constraint*, *degree of freedom (DoF)*, *directionality, disassembled joint*, *follower (follower body)*, *ground*, *inertia tensor, massless connector*, *primitive joint*, and *sensor*.

kinematics

A *kinematic* analysis of a mechanical system specifies topology, degrees of freedom (DoFs), motions, and constraints, without specification of applied forces/torques or the mass properties of the bodies.

The *machine state* at some time is the set of all

- Instantaneous positions and orientations
- Instantaneous velocities

of all bodies in the system, for both linear (translational) and angular (rotational) DoFs of the bodies.

Specification of applied forces/torques and solution of the system's motion as a function of time are given by the system's dynamics.

See also *constraint*, *degree of freedom (DoF)*, *dynamics*, and *topology*.

local CS

A *local coordinate system (CS)* is attached to either a Ground or a Body:

- A *Grounded CS* is automatically defined when you represent a ground point by a Ground block and is always at rest in the World reference frame. The origin of this Grounded CS is the same point as the ground point and not in general the same as the World CS origin.
- You define one or more *Body CSs* when you configure the properties of a Body. A Body CS is fixed rigidly in the body and carried along with that body's motion.

  To indicate an attached coordinate system, a Body block has an axis triad CS port in place of the open, round connector port.

See also *body*, *Body CS*, *coordinate system (CS)*, *grounded CS*, *reference frame (RF)*, and *World*.

machine

In a SimMechanics model, a *machine* is a complete, connected block diagram representing one mechanical system. It is topologically isolated from any other machine in your model and has at least one ground and exactly one Machine Environment block.

A SimMechanics model has one or more machines.

See also *ground* and *topology*.

machine precision constraint

A *machine precision constraint* implements motion restrictions on constrained degrees of freedom to the precision of your computer processor's arithmetic. It is the most robust, computationally intensive, and slowest-simulating constraint.

The precision to which the constraint is maintained depends on scale or the physical system of units.

See also *constraint, stabilizing constraint,* and *tolerancing constraint.*

mass

The *mass* is the proportionality between a force on a body and the resulting translational acceleration of that body.

Let V be the body's volume and $\rho(\boldsymbol{r})$ its mass density, a function of position $\boldsymbol{r}$ within the body. Then the mass m is:

$$m = \int_V dV\, \rho(\boldsymbol{r})$$

The mass is a real, positive scalar or equivalent MATLAB expression.

A body's mass is insensitive to choice of reference frame, coordinate system origin, or coordinate axes orientation.

See also *body* and *inertia tensor*.

massless connector

A *massless connector* is a machine component equivalent to two joints whose respective primitive axes are spatially separated by a fixed distance. You can specify the gap distance and the axis of separation. The space between the degrees of freedom is filled by a rigid connector of zero mass.

You cannot actuate or sense a massless connector.

See also *disassembled joint* and *joint*.

open machine

You can disconnect an *open machine* diagram into two separate diagrams by cutting no more than one joint.

Such machines can be divided into two types:

- An *open chain* is a series of bodies connected by joints and topologically equivalent to a line.

- An *open tree* is a series of bodies connected by joints in which at least one body has more than two joints connected to it. Bodies with more than two connected joints define *branch points* in the tree. A tree can be disconnected into multiple chains by cutting the branch points.

The end body of a chain is a body with only one connected joint.

See also *closed loop machine* and *topology*.

part

A *part* represents a body in a computer-aided design (CAD) assembly. In CAD representations, a part typically includes full geometric, as well as mass and inertia tensor, information about a body.

Body degrees of freedom are restricted in CAD by constraints (sometimes called *mates*).

After translation into a SimMechanics model, parts are represented by bodies.

See also *assembly*, *associativity*, *body*, *computer-aided design (CAD)*, *constraint*, *degree of freedom (DoF)*, *inertia tensor*, and *mass*.

physical tree

You obtain the *physical tree* representation from a full machine diagram by removing actuators and sensors and cutting each closed loop once. The physical tree retains bodies, joints, constraints, and drivers.

See also *closed loop machine, open machine, spanning tree,* and *topology*.

primitive joint

A *primitive joint* expresses one degree of freedom (DoF) or coordinate of motion, if this DoF is a translation along one direction (*prismatic joint*) or a rotation about one fixed axis (*revolute joint*).

A SimMechanics *spherical joint* (three DoFs: two rotations to specify directional axis, one rotation about that axis) is also treated as a primitive joint.

These three types of primitive joints are the *joint primitives* from which composite joints are built.

A weld primitive has no degrees of freedom.

See also *composite joint* and *joint*.

principal axes

The inertia tensor of a body is real and symmetric and thus can be diagonalized, with three real eigenvalues and three orthogonal eigenvectors. The *principal axes* of a body are these eigenvectors.

See also *equivalent ellipsoid*, *inertia tensor*, and *principal inertia moments*.

principal inertial moments

The inertia tensor of a body is real, symmetric, and diagonalizable, with three real eigenvalues and three orthogonal eigenvectors. The *principal inertial moments* or *principal moments of inertia* of a body are these eigenvalues, the diagonal values when the tensor is diagonalized.

The principal moments of a real body satisfy the *triangle inequalities:* the sum of any two moments is greater than or equal to the third moment.

If two of the three principal moments are equal, the body has some symmetry and is dynamically equivalent to a *symmetric top*. If all three principal moments are equal, the body is dynamically equivalent to a *sphere*.

See also *equivalent ellipsoid*, *inertia tensor*, and *principal axes*.

quaternion

A *quaternion* mathematically represents a three-dimensional spherical rotation as a four-component row vector of unit length:

$$q = \left[n_x \sin(\theta/2),\ n_y \sin(\theta/2),\ n_z \sin(\theta/2),\ \cos(\theta/2) \right] = \left[\boldsymbol{q}_{\mathrm{v}}, q_{\mathrm{s}} \right]$$

with $q*q = 1$. The vector $\boldsymbol{n} = (n_x, n_y, n_z)$ is a three-component vector of unit length: $\boldsymbol{n} \cdot \boldsymbol{n} = 1$. The unit vector $\boldsymbol{n}$ specifies the axis of rotation. The rotation angle about that axis is θ and follows the right-hand rule.

The axis-angle representation of the rotation is just [$\boldsymbol{n}$ θ].

See also *axis-angle rotation*, *degree of freedom (DoF)*, *Euler angles*, *primitive joint*, *right-hand rule*, and *rotation matrix*.

reference frame (RF)

A *reference frame (RF)* is the state of motion of an observer.

An *inertial* RF is a member of a set of all RFs moving uniformly with respect to one another, without relative acceleration. This set defines inertial space.

An RF is necessary but not sufficient to define a coordinate system (CS). A CS requires an origin point and an oriented set of three orthogonal axes.

See also *coordinate system (CS)*, *local CS*, and *World*.

RF

A *reference frame (RF)*.

right-hand rule

The *right-hand rule* is the standard convention for determining the sign of a rotation: point your right thumb into the positive rotation axis and curl your fingers into the forward rotational direction.

See also *degree of freedom (DoF), directionality,* and *joint*.

root body

A *root body* is a component of a SimMechanics model translated from a computer-aided design (CAD) assembly. In the translated model, a block sequence Ground — Root Weld — Root Body or Root Body — Root Weld — Fixed Body represents the CAD assembly's fundamental or subassembly root, respectively.

In CAD assemblies, the fundamental or subassembly root represents a fixed point relative to which all part motion or subassembly part motion is measured. The fundamental root is usually the same as the assembly origin.

See also *assembly*, *body*, *computer-aided design (CAD)*, *fixed part*, *fundamental root*, *ground*, *subassembly*, and *fundamental root*.

rotation matrix

A *rotation matrix* mathematically represents a three-dimensional spherical rotation as a 3-by-3 real, orthogonal matrix R: $R^{\mathrm{T}}R = RR^{\mathrm{T}} = I$, where I is the 3-by-3 identity and R^{T} is the transpose of R.

$$R = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} = \begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix}$$

In general, R requires three independent angles to specify the rotation fully. There are many ways to represent the three independent angles. Here are two:

- You can form three independent rotation matrices R_1, R_2, R_3, each representing a single independent rotation. Then compose the full rotation matrix R with respect to fixed coordinate axes (like World) as a product of these three: $R = R_3 * R_2 * R_1$. The three angles are *Euler angles*.
- You can represent R in terms of an *axis-angle rotation* $\boldsymbol{n} = (n_x, n_y, n_z)$ and θ with $\boldsymbol{n} \cdot \boldsymbol{n} = 1$. The three independent angles are θ and the two needed to orient $\boldsymbol{n}$. Form the antisymmetric matrix:

$$J = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}$$

Then *Rodrigues' formula* simplifies R:

$$R = \exp(\theta J) \; = \; I \; + \; J \sin\theta \; + \; J^2(1 - \cos\theta)$$

See also *axis-angle rotation*, *degree of freedom (DoF)*, *Euler angles*, *primitive joint*, *quaternion*, and *right-hand rule*.

sensor

A *sensor* is a machine component that measures the motion of, or forces/torques acting on, a body or joint. A sensor can also measure the reaction forces in a constraint or driver constraining a pair of bodies.

A SimMechanics Sensor block has an open round SimMechanics connector port ○ for connecting with a Body or Joint block and an angle bracket > Simulink outport for connecting with normal Simulink blocks, such as a Sinks block like Scope.

See also *actuator*, *body*, *connector port*, *constraint*, *driver*, *joint*, and *primitive joint*.

spanning tree

You obtain the *spanning tree* representation from a full machine diagram by removing everything except bodies and joints and cutting each closed loop once.

See also *closed loop machine, open machine, physical tree,* and *topology*.

stabilizing constraint

A *stabilizing constraint* implements motion restrictions on constrained degrees of freedom by modifying the dynamics of a system so that the constraint manifold is attractive, without changing the constrained solution. This constraint solver type is computationally the least intensive, the least robust, and the fastest-simulating.

The precision to which the constraint is maintained depends on scale or the physical system of units.

See also *constraint*, *machine precision constraint*, and *tolerancing constraint*.

stiction actuator

A *stiction actuator* is a machine component that applies discontinuous friction forces to a joint primitive according to the relative velocity of one body with the other body.

If this relative velocity drops below a specified threshold, the relative motion ceases and the bodies or joints become locked rigidly to one another by static friction.

Above that threshold, the bodies or joints move relative to one another with kinetic friction.

See also *actuator, composite joint, dynamics, joint,* and *primitive joint*.

STL

Stereolithographic (STL) format is an open file format for specifying the three-dimensional surface geometry or shape of a body. STL format specifies surface geometry by linked triangles whose edges and vertices are oriented by the right-hand rule.

SimMechanics visualization uses STL format files for custom body geometries and supports both ASCII and binary STL variants.

See also *body*, *computer-aided design (CAD)*, *convex hull*, *equivalent ellipsoid*, *part*, *right-hand rule*, *VRML* and 3D Systems, creator of STL format, at `www.3dsystems.com`.

subassembly

A *subassembly* represents of a subset of machine parts and constraints (mates) in computer-aided design (CAD).

A subassembly is attached to its parent CAD assembly at a single branching point.

A subassembly is either *flexible* or *rigid*. That is, its parts either can move with respect to one another, or they cannot.

See also *assembly*, *computer-aided design (CAD)*, *constraint*, *part*, and *subassembly root*.

subassembly root

A *subassembly root* is a point in a computer-aided design (CAD) subassembly that does not move relative to the assembly point off of which it branches. All translational and rotational motion of parts in the subassembly reference this unmoving point.

See also *assembly*, *computer-aided design (CAD)*, *fundamental root*, *part*, *root body*, and *subassembly*.

tolerancing constraint

A *tolerancing constraint* implements motion restrictions on constrained degrees of freedom only up to a specified accuracy and/or precision.

This accuracy/precision is independent of any accuracy/precision limits on the solver used to integrate the system's motion, although constraints cannot be maintained to greater accuracy than the accuracy of the solver.

The precision to which the constraint is maintained depends on scale or the physical system of units.

Tolerancing constraints are moderately robust and moderately intensive and execute at moderate speed. They are less intensive than

machine precision constraints, but computationally more intensive than stabilizing constraints.

Tolerancing constraints are most useful in realistic simulation of constraint slippage (“slop” or “play”).

See also *constraint*, *machine precision constraint*, and *stabilizing constraint*.

topology

The *topology* of a machine diagram is the global connectivity of all its elements. A diagram’s *elements* are its bodies, and its *connections* are its joints, constraints, and drivers. Two topologies are *equivalent* if you can transform one diagram into another by continuous deformations and without cutting connections or joining elements.

An *open machine* diagram has no closed loops.

- An *open chain* is topologically equivalent to a line; and each body is connected to only two other bodies, if the body is internal, or one other body if it is at an end.
- An *open tree* has one or more *branch points*. A branch point is where an internal body is connected to more than two joints. A tree can be disconnected into multiple chains by cutting at the branch points.

A *closed loop machine* diagram has one or more closed loops. The number of closed loops is equal to the minimum number of joints, minus one, that must be cut to dissociate a diagram into two disconnected diagrams.

An actual diagram can have one of these primitive topologies or can be built from multiple primitive topologies.

See also *body*, *closed loop machine, constraint*, *driver*, *joint, machine*, *open machine*, and *spanning tree*.

VRML

Virtual Reality Modeling Language (VRML) is an open, Web-oriented ISO standard for defining three-dimensional virtual worlds and bodies in multimedia and the Internet.

In VRML, body rotations are represented in the axis-angle form. The SimMechanics RotationMatrix2VR block converts rotation matrices to the equivalent axis-angle forms.

See also *axis-angle rotation*, *STL*, and the Web3D Consortium at `www.web3d.org`.

World

In the SimMechanics environment, *World* is a kinematic and geometric construct defining both the absolute inertial *reference frame (RF)* and absolute *coordinate system (CS)* in that RF. World has a fixed origin and fixed coordinate axes that cannot be changed.

The World coordinate axes are defined so that:

+*x* points right

+*y* points up (gravity in -*y* direction)

+*z* points out of the screen, in three dimensions

The assembly origin of a CAD assembly translated into a model becomes the World CS origin.

See also *adjoining CS*, *computer-aided design (CAD)*, *coordinate system (CS)*, *fundamental root, ground, grounded CS,* and *reference frame (RF).*